JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 12, Number 4, 2005
© Mary Ann Liebert, Inc.
Pp. 416–430

sFFT: A Faster Accurate Computation of the p-Value of the Entropy Score

URI KEICH

ABSTRACT

We present sFFT, an algorithm for efficiently computing the p-value of the information content, or the entropy score of an alignment of DNA sequences. Applying the FFT algorithm to an exponentially shifted probability mass function allows us perform fast convolutions that do not suffer from the otherwise overwhelming effect of accumulated numerical roundoff errors. Through a rigorous analysis of the propagation of numerical errors across the various steps of sFFT, we provide a theoretical bound on the overall error of our computed p-value. The accuracy of the computed p-value, as well as the utility of the error bound, are empirically demonstrated. Although there are faster algorithms that would compute this p-value, they can err significantly; sFFT is the fastest reliable algorithm. Finally, we note that the basic algorithm is likely to be applicable in a wider context than the one considered here.

Key words: information content, entropy score, p-value, FFT, numerical errors, large deviation, exponential shifted measure.

1. INTRODUCTION

Aligning biologically related sequences is a common task in computational biology. Such alignments arise, for example, in studies of transcription factor binding sites. Typically, such a study starts with an input set of DNA sequences which are believed to contain the a priori unknown and relatively short binding sites. Although these sites are not identical, they are presumably "fairly similar" to one another in a way that would be clearer once we align all these instances. In order to analyze the alignment, one usually constructs the alignment matrix which lists the number of occurrences of each letter in each of the columns, or the positions, of the alignment.

Although an alignment matrix does shed some light on the quality of the alignment, one usually tries to "summarize" the matrix by assigning it a score. One such figure of merit is the commonly used information content, or entropy score (Stormo, 2000). Let n_{ij} denote the number of occurrences of the jth letter in the ith column, and let $n = \sum_j n_{ij}$ be the number of instances, or sequences, in the alignment. The entropy score of the motif is

$$I := \sum_i \sum_j n_{ij} \log \frac{n_{ij}/n}{q_j},$$

where q_j is the background frequency of the jth letter (typically its frequency in the given sample).

Computer Science Department, Cornell University, Ithaca, NY 14853.

Having computed the score, we are interested in assessing its statistical significance. This is often accomplished by computing the p-value of the observed score. Let $I(i) = \sum_j n_{ij} \log\left[(n_{ij}/n)/q_j\right]$ be the entropy of the ith column. Then, $I(i)$ is also the generalized log likelihood ratio between the hypothesis $H_0 = H_0(i)$ that $\{n_{ij}\}_j$ is a sample of the multinomial distribution specified by the background frequencies $\{q_1, \dots, q_A\}$, and between the alternative hypothesis H_1 that the sample originated from *some* multinomial distribution on A letters. It is well known that under H_0, $2I(i) \to \chi^2(A-1)$ in distribution, as $n \to \infty$ (e.g., Rice [1995]). Therefore, under the hypothesis $H_0 = \cap_i H_0(i)$, and assuming the L columns of the alignment are independent of one another, $2I \to \chi^2(L(A-1))$. The problem, however, is that in a typical application of motif finding we are interested in extreme cases where many of the letters in each column rarely appear, if at all, in which case the above asymptotic analysis is not applicable.

At this point, we would benefit from a mathematical representation of the problem. Let p denote the pmf (probability mass function) of $I(i)$ under H_0. Assuming that for $i = 1, \dots, L$, $I(i)$ are independent random variables, the pmf of the entropy score, $I = \sum_1^L I(i)$, is given by the L-fold convolution of p:

$$p^{*L}(s) := \underbrace{p * \cdots * p}_{L} := \sum_{\substack{(s_1,\dots,s_L): \\ s_1+\cdots+s_L=s}} p(s_1)\dots p(s_L). \tag{1}$$

Define

$$\bar{F}^{*L}(s) := \sum_{s' \geq s} p^{*L}(s'); \tag{2}$$

then the p-value of $I = s_0$ is $\bar{F}^{*L}(s_0)$. In particular, to compute this p-value, we need to compute $p^{*L}(s)$ for all $s \geq s_0$, which would typically require a prohibitive amount of time. Having realized that fact in a more general context, Staden (1989) suggested replacing I with a lattice-discretized version of it. For example, we can replace $I(i)$ with $I_\delta(i) = h_\delta(I(i))$, where

$$h_\delta(x) := \lfloor x/\delta \rfloor \cdot \delta. \tag{3}$$

Then, $I_\delta = \sum_i I_\delta(i)$ is a lattice approximation of I. The pmf of I_δ is given by p_δ^{*L}, the L-fold convolution of p_δ, the pmf of $I_\delta(i)$. The reason p_δ^{*L} is much easier to compute than the original p^{*L} is that it is a convolution on a lattice.

Hertz and Stormo (1999) suggest two ways to estimate $\bar{F}^{*L}(s_0)$. The first method, **NC**, is based on Staden's aforementioned technique of approximating $\bar{F}^{*L}(s_0)$ with

$$\bar{F}_\delta^{*L}(s_0) := \sum_{s \geq s_0} p_\delta^{*L}(s). \tag{4}$$

For computing p_δ, they suggest a dynamic programming algorithm that computes the lattice pmf in $O(AMn^2)$, where A is the size of the alphabet, n is the number of instances or sequences, and M is the size of the lattice or the maximal possible number of discretized values.

The second method Hertz and Stormo suggest, LD, involves large deviation theory. More precisely, LD which is implemented as part of their motif finder Consensus (Hertz and Stormo, 1999), uses one of three distributions (gamma, normal, reversed gamma) to estimate the tail of the exponentially shifted probability measure. Which one of the three distributions is actually used is determined by the distance of the observed value of the entropy score, s_0, from the maximal/minimal possible scores. In practice, their approximation scheme works quite well except for a range of values near the maximum possible score, where this method can be off by a significant factor (see Section 5 for an example where the LD p-value is off by a factor of over 100 from the correct answer).

Comparing the two methods, the authors consider NC as the more reliable one though they do not attempt to quantify that. Another advantage NC enjoys is that it actually yields the entire distribution of the entropy score as opposed to merely the p-value of s_0. However, these advantages often pale given that NC is significantly slower than LD: for example, with $A = 4$, $L = 10$, and $n = 100$, it is 170 times slower and other settings produce even significantly worse results for NC (Hertz and Stormo, 1999).

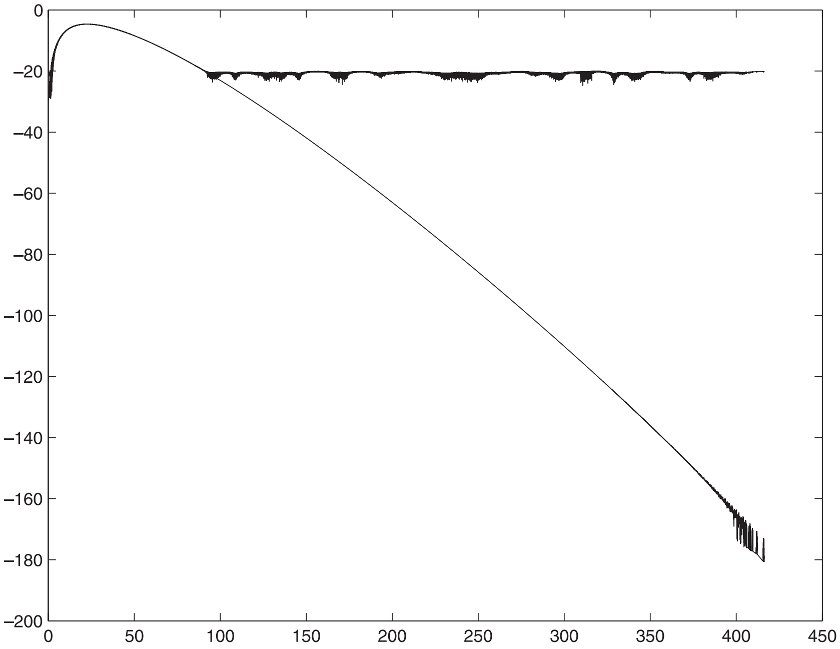

FIG. 1. Overwhelming errors in an unshifted FFT-based convolution. The figure illustrates the potential overwhelming effects of numerical errors in applications of (unshifted) FFT. Compared are the graphs of two estimates of the p-value $\bar{F}_\delta^{*L}(s)$. The smoother graph is of the provably reliable NC, while the second is obtained from FFT-based convolution applied to p_δ. Both graphs are $\log_{10}$ of the function. Note that in this case the FFT-based estimate fails to capture p-values smaller than roughly 10^{-21}.

Remark. A third method for estimating the p-value, based on Fisher's "omnibus" procedure (Bailey and Gribskov, 1998), is implemented in the latest release (3.0) of MEME (Bailey and Elkan, 1994). We found this method to be considerably less accurate than the first two (Jones *et al.*, in preparation).

As far as the complexity of their NC algorithm, Hertz and Stormo note that the complexity of computing p_δ^{*L} using a naive convolution is $O(L^2M^2)$. They also comment that implementing this convolution using the FFT (fast Fourier transform, e.g., Press *et al.* [1992]) would reduce the complexity to $O(LM\log(LM))$. However, they do not report any implementation of this strategy, most likely because the accumulated effects of roundoff errors mask out the smaller (extreme) values of the pmf. See Fig. 1 for an example of the havoc wreaked by the accumulation of roundoff errors in FFT-based convolution. Notice in particular how all the smaller p-values, which are typically the ones we are interested in, are completely corrupted.

In this paper we present sFFT, an algorithm that incorporates the large-deviation technique of exponential shift into FFT-based convolution. This allows sFFT to share the reduced complexity of the FFT and, at the same time, avoid the tendency of the FFT to accumulate dominating round-off errors. We begin by presenting our method and continue to provide empirical and theoretical evidence for the validity of our approach.

2. DFT, FFT, AND LARGE DEVIATION

First let us briefly recall how one convolves using DFT (discrete Fourier transform). The L-fold convolution of a vector $v \in \mathbb{C}^M$, v^{*L}, can be computed as follows. Let $N = ML$ and extend v as an N-dimensional vector by padding it with zeroes. Then compute Dv, the DFT of v, which is defined (Press *et al.*, 1992) as follows:

$$\hat{v}(k) := (Dv)(k) := \sum_{j=0}^{N-1} e^{ijk\omega} v(j),$$

where $\omega = 2\pi/N$, $D = D(N)$, and $i = \sqrt{-1}$. Define $w \in \mathbb{C}^N$ as $w(k) = \left[\hat{v}(k)\right]^L$. Then, v^{*L} is given by the inverse DFT of w:

$$v^{*L}(l) = (D^{-1}w)(l) := \frac{1}{N}\sum_{j=0}^{N-1} e^{-ijl\omega} w(j).$$

Let $\widetilde{D}$ and $\widetilde{D^{-1}}$ be the FFT implemented[1] machine operators which serve as our surrogates for the DFT operators D and D^{-1}. Note that $\widetilde{D}$ and $\widetilde{D^{-1}}$ are not exactly the linear and mutually inverse operators that D and D^{-1} are. The reason FFT-based convolution typically fails in our context is that it relies on delicate cancellations to produce the final result: $(D^{-1}w)(l)$ is a sum of N complex numbers. In the example we study in Section 5, the magnitudes of these numbers roughly span a range of values from 10^{-5} to 3×10^{-70}. Since ε_*, the relative machine precision, is roughly 10^{-16}, we cannot expect in this case much accuracy from $\widetilde{D^{-1}}$ beyond 10^{-21} (as can be verified in Fig. 1).

To avoid the problem of roundoff errors dominating the smaller p-values, we adopt an idea from large deviation theory. Namely, we apply an exponential shift to our pmf p_δ prior to convolving it, thereby amplifying the entries we are interested in. Let

$$p_{\theta,\delta}(s) := p_\delta(s)e^{\theta s}/M_\delta(\theta), \tag{5}$$

where $M_\delta(\theta) = Ee^{\theta I_\delta(i)}$ is the moment-generating function of the lattice score of one column.

The reason we choose an exponential shift is that it essentially commutes with the convolution operator. More precisely, from (1):

$$\begin{aligned} p_{\theta,\delta}^{*L}(s)e^{-\theta s + L\log M_\delta(\theta)} &= \sum_{\substack{(s_1,\dots,s_L):\\ s_1+\dots+s_L=s}} p_{\theta,\delta}(s_1)\dots p_{\theta,\delta}(s_L)e^{-\theta s+L\log M_\delta(\theta)} \\ &= \sum_{\substack{(s_1,\dots,s_L):\\ s_1+\dots+s_L=s}} p_\delta(s_1)e^{\theta s_1 - \log M_\delta(\theta)}\dots p_\delta(s_L)e^{\theta s_L - \log M_\delta(\theta)}e^{-\theta s+L\log M_\delta(\theta)} \\ &= p_\delta^{*L}(s). \end{aligned}$$

The obvious next question is how should we choose the parameter θ. An intuitively appealing choice of θ is the one which comes from large deviation theory. That is, given s_0, we choose

$$\theta_0 = \operatorname{argmin}_\theta \left[\log M(\theta) - \theta s_0/L\right]. \tag{6}$$

This choice of θ roughly centers the shifted distribution of one column about s_0/L, or more precisely, the mean of p_{δ,θ_0} is s_0/L. Thus, p_{δ,θ_0}^{*L} is centered about s_0 and therefore has maximal "resolving power" (relative to the "noise" of numerical errors) around s_0. Intuitively, the significant contributions to the p-value should come from values of p_δ^{*L} close to s_0, and therefore it looks as a sensible choice. In the following, we present empirical and theoretical justification for this particular choice of θ.

3. sFFT

For a vector $x \in \mathbb{C}^N$, we denote by $\widetilde{x} \in \mathbb{C}^N$ its machine representation, or more generally, in case some calculations are involved, its machine approximation.[2] In the latter case, we assume the calculations are given implicitly.

[1] The FFT algorithm is a recursive implementation of DFT that reduces the complexity from $O(N^2)$ to $O(N\log N)$.

[2] Throughout this paper we refer to an approximated, or estimated values as opposed to the exact ones due to the presence of roundoff errors, see e.g. Stoer and Bulirsch (1992).

3.1. *The algorithm*

The input to sFFT is

- n, the number of sequences,
- L, the number of columns in the alignment,
- $q_1, \dots, q_A$, the background frequencies of the A letters,
- M, the size of the lattice,
- s_0, the observed score.

Given the input, sFFT does the following:

1. Computes $\widetilde{p}_\delta$, an estimate of p_δ. Currently this is implemented by naively enumerating all possible observed frequency tables.
2. Finds θ_0 by numerically solving (6). Technically, we implemented that by applying the procedures mnbrak followed by dbrent of Press *et al.* (1992).
3. Computes $\widetilde{p_{\theta_0,\delta}}(s)$ according to (5).
4. Computes $\widetilde{p^{*L}_{\theta_0,\delta}}$ by applying the FFT-based convolution as described in Section 2 to $\widetilde{p_{\theta_0,\delta}}(s)$ (we identify a lattice pmf with the obvious associated vector).
5. For all $s \geq s_0$, computes $\widetilde{p^{*L}_\delta}(s) = \widetilde{p^{*L}_{\theta_0,\delta}}(s) e^{-\theta_0 s + L \log \widetilde{M}_\delta(\theta_0)}$.
6. Returns sFFT(s_0) $:= \sum_{s \geq s_0} \widetilde{p^{*L}_\delta}(s)$.

3.2. *Complexity and extensions*

Our current implementation of the first step is not very efficient. Its complexity is $O(M + n^{A-1})$ which is feasible for a typical application of motif finding in DNA ($A = 4$ and $n \lesssim 100$). Alternatively, one can use Hertz and Stormo's (1999) dynamic programming approach ($O(AMn^2)$). Strictly speaking, this would require reworking of some of the error estimates we provide below.

The complexity of numerically solving (6) for θ_0 is typically negligible. Indeed, dbrent (Press *et al.*, 1992), which essentially implements Brent's iterative method for minimization, has at least linear and typically superlinear convergence (in the number of required precision digits). Finally, the complexity of each iterative step is $O(M)$ and the number of digits of precision should be no more than $-\log_{10}\sqrt{\varepsilon_*} \approx 8$.

With $N = ML$, the complexity of the FFT-based convolution is $O(LM\log(LM))$. Assuming that typically $M \gg n$, this term would dominate the overall complexity provided we limit ourselves to a small A (as in DNA sequences). For larger A, a more careful implementation and analysis of Hertz and Stormo's algorithm for step 1 is required.

Note that we can greatly extend the applicability of our algorithm by working with logarithms in steps 1, 2, 5, and 6 of the algorithm. This would not change the complexity but would allow us to estimate p-values that would otherwise create an underflow (roughly 10^{-308}). Such an extension might be necessary when dealing with a reasonably large sample of DNA sequences.

Finally we will use the following trivial extensions of sFFT:

- We can add an explicitly desired shift θ_0 as an input parameter and skip step 2. We will denote such cases as sFFT(s_0, θ_0) (the other parameters being implicit).
- We can stop at step 5 and return $\widetilde{p^{*L}_\delta}$, an estimator of the pmf p^{*L}_δ. We will denote such cases as sFFT(θ_0).

4. QUALITY CONTROL

In this section, we establish upper bounds on the error in estimating the p-value of $I = s_0$. We do so for both Hertz and Stormo's NC algorithm and our sFFT. Both algorithms suffer from errors induced by the same two sources. The first type, the lattice errors, is due to the fact that we only consider I_δ, the latticed version of the original entropy function I. The second type of errors is numerical errors – our algorithms are implemented in finite precision machines. In Section 4.1, we discuss the lattice-induced errors and how we can bound them. In Section 4.2, we briefly study numerical errors in general. Section 4.3 analyzes the numerical errors of Hertz and Stormo's NC algorithm, and finally Section 4.4 studies those in the case of sFFT.

4.1. Lattice errors

Recall that we only use a lattice in order to facilitate the estimation of $\bar{F}^{*L}(s_0)$, (2). Although we can simply approximate the latter with $\bar{F}_\delta^{*L}(s_0)$, (4), in this section we are interested in how much we can err in doing so. The next claim shows how to bound $\bar{F}^{*L}(s_0)$ given $\bar{F}_\delta^{*L}(s)$.

Claim 1. *For $j\delta \le s < (j+1)\delta$, let*

$$\begin{aligned} UB_\delta(s) &:= \bar{F}_\delta^{*L}((j+1-L)\delta) \\ LB_\delta(s) &:= \begin{cases} \bar{F}_\delta^{*L}(j\delta) & s = j\delta \\ \bar{F}_\delta^{*L}((j+1)\delta) & s > j\delta \end{cases}. \end{aligned} \tag{7}$$

Then

$$LB_\delta(s) \le \bar{F}^{*L}(s) \le UB_\delta(s). \tag{8}$$

Proof. We first prove the claim for $s = j\delta$. For $\boldsymbol{k} \in \mathbb{N}^L$, let $\|\boldsymbol{k}\| = \sum_1^L k_i$. From (1) and the definition of p_δ,

$$p_\delta^{*L}(j\delta) = \sum_{\boldsymbol{k}:\|\boldsymbol{k}\|=j} \prod_{i=1}^{L} \sum_{s_i:h_\delta(s_i)=k_i\delta} p(s_i) = \sum_{\substack{(s_1,\ldots,s_L):\\ \sum_1^L h_\delta(s_i)=j\delta}} \prod_{i=1}^{L} p(s_i). \tag{9}$$

It follows from (3) that $0 \le x - h_\delta(x) < \delta$, and therefore

$$0 \le h_\delta\left(\sum_{i=1}^{L}(s_i - h_\delta(s_i))\right) \le (L-1)\delta.$$

Hence,

$$\sum_{i=1}^{L} h_\delta(s_i) \le h_\delta\left(\sum_{i=1}^{L} s_i\right) \le \sum_{i=1}^{L} h_\delta(s_i) + (L-1)\delta.$$

Since $p(s_i) \ge 0$, it follows that

$$\sum_{\substack{(s_1,\ldots,s_L):\\ \sum_1^L h_\delta(s_i)\ge j\delta}} \prod_{i=1}^{L} p(s_i) \le \sum_{\substack{(s_1,\ldots,s_L):\\ h_\delta\left(\sum_1^L s_i\right)\ge j\delta}} \prod_{i=1}^{L} p(s_i) \le \sum_{\substack{(s_1,\ldots,s_L):\\ \sum_1^L h_\delta(s_i)+(L-1)\delta\ge j\delta}} \prod_{i=1}^{L} p(s_i). \tag{10}$$

Combining (4), (7), (9), and (10) proves (8) for $s = j\delta$.

Finally, for $j\delta \le s < (j+1)\delta$, using what we just proved for lattice points and (7),

$$\bar{F}^{*L}(s) \ge \bar{F}^{*L}((j+1)\delta) \ge LB_\delta((j+1)\delta) = \bar{F}_\delta^{*L}((j+1)\delta),$$

and similarly,

$$\bar{F}^{*L}(s) \le \bar{F}^{*L}(j\delta) \le UB_\delta(j\delta) = \bar{F}_\delta^{*L}((j+1-L)\delta). \qquad \blacksquare$$

Remarks.

- Currently, $\text{sFFT}(j\delta) = \widetilde{LB}_\delta(j\delta)$ though this is somewhat arbitrary. Estimating $\bar{F}^{*L}(s)$ with any number in $[LB_\delta(s), UB_\delta(s)]$ yields an error bounded by $UB_\delta(s) - LB_\delta(s)$. In practice, there is little difference between those two bounds except perhaps near the maximal possible score.
- Generally speaking, as δ decreases (by increasing M) $[LB_\delta(s), UB_\delta(s)]$, the bounding interval for $\bar{F}^{*L}(s)$ shrinks. Strictly speaking, though, this is only guaranteed to be the case if the new lattice is a refinement of the previous one.
- In reality, we can only compute $\widetilde{\bar{F}_\delta^{*L}}$, a machine computable estimate of $\bar{F}_\delta^{*L}$, and the relative error between the two will increase with M. In practical terms, though, as long as this relative error is reasonably small, we can just plug $\widetilde{\bar{F}_\delta^{*L}}$ in (7) instead of $\bar{F}_\delta^{*L}$. We can thus obtain an estimate of the lattice error.

4.2. A brief introduction to numerical errors

Numerical errors arise from the finiteness of our machine's precision. More precisely, there exists $\varepsilon_* > 0$, typically $\varepsilon_* \approx 10^{-16}$, such that for $|a| < \varepsilon_*$, $\widetilde{1+a} = 1$. Hence, the *relative error* $(|\widetilde{x} - x|/x)$ in computing $1 + a$ can be arbitrarily close to ε_*. While such an error is rarely a cause for concern, the propagation or accumulation of such small errors can be disastrous as is evident in Fig. 1. In the remainder of this section, we spell out the basic technical results we need to allow us to track the propagation of numerical errors in the algorithms we analyze. The executive summary of this section is as follows:

- The relative error of a sum of n positive numbers equals the maximal error among the input numbers plus $n\varepsilon_*$.
- The relative error of a product of two real numbers equals the sum of their errors.

First, we assume that our machine does not introduce an error bigger than necessary when performing elementary calculations. More precisely, let $x, y \in \mathbb{R}$ with $x = \widetilde{x}$ and $y = \widetilde{y}$; then we have the following:

$$\widetilde{x+y} - (x+y) = \varepsilon_{x+y}(x+y) \qquad \text{where } |\varepsilon_{x+y}| < \varepsilon_* \tag{11}$$

$$\widetilde{xy} - xy = \varepsilon_{xy}(xy) \qquad \text{where } |\varepsilon_{xy}| < \varepsilon_* \tag{12}$$

These assumptions hold when x and y are perfectly estimated. The question now is how are errors in estimating x and y propagated.

Lemma 1. *Assume that $a, b \geq 0$ and that for $x = a, b$: $\widetilde{x} = x + e_x$ with $|e_x| \leq x(c_x\varepsilon_* + d_x\varepsilon_*^2)$, with $c_x, d_x \geq 0$. Then, with $C = \max\{c_a, c_b\}$ and $D = \max\{d_a, d_b\}$,*

$$|e_{a+b}| := \left|\widetilde{\widetilde{a} + \widetilde{b}} - (a+b)\right| \leq (a+b)\left[(1+C)\varepsilon_* + (C+D)\varepsilon_*^2 + D\varepsilon_*^3\right].$$

Proof. From (11) and our definitions above with $|\varepsilon_{\widetilde{a}+\widetilde{b}}| < \varepsilon_*$,

$$\begin{aligned} e_{a+b} &= [(a+e_a) + (b+e_b)]\,(1+\varepsilon_{\widetilde{a}+\widetilde{b}}) - (a+b) \\ &= e_a + e_b + (a+b)\varepsilon_{\widetilde{a}+\widetilde{b}} + (e_a+e_b)\varepsilon_{\widetilde{a}+\widetilde{b}}. \end{aligned} \tag{13}$$

The result follows by elementary arithmetic. ■

Lemma 2. *Suppose $a_i \geq 0$ and $\widetilde{a_i} = a_i + e_i$, where $|e_i| \leq a_i c\varepsilon_*$ with $c > 0$. Then, with $S_n = \sum_1^n a_i$ and assuming $(n+c)\varepsilon_* < 1$,*

$$\left|\widetilde{\sum_1^n \widetilde{a_i}} - S_n\right| \leq S_n(n+c)\varepsilon_*(1+n\varepsilon_*). \tag{14}$$

Remark. In most practical applications, $(n+c)\varepsilon_* \ll 1$ and the lemma is applicable. Since in particular $n\varepsilon_* \ll 1$, the lemma says that the relative error in estimating S_n grows linearly with n regardless of how inaccurate were the estimates $\tilde{a}_i$.

Proof. We prove by induction that if the mild condition $(n + c)\varepsilon_* < 1$ holds then so does (14). For $n = 1$, (14) holds by the assumptions on a_1. For the inductive step, use the previous lemma with $a = S_{n-1}$ and $b = a_n$, where c_a and d_a are provided by the inductive argument and $c_b = c$, $d_b = 0$. It is helpful to note that

$$S_n(n + c)\varepsilon_*(1 + n\varepsilon_*) = S_n\left[(n + c)\varepsilon_* + (n^2 + nc)\varepsilon_*^2\right]. \qquad \blacksquare$$

The next lemma shows that multiplication is not as well behaved as addition of positive numbers.

Lemma 3. *Suppose a and b are as in Lemma 1. Then,*

$$|e_{ab}| := \left|\widetilde{\tilde{a}\tilde{b}} - ab\right| \le ab\left[(1 + c_a + c_b)\varepsilon_* + (c_a + c_b + c_a c_b)\varepsilon_*^2 + c_a c_b \varepsilon_*^3\right].$$

Proof. The result follows immediately from the following equation which can be derived from (12) and the above definitions:

$$e_{ab} = [(a + e_b)(b + e_b)]\,(1 + \varepsilon_{\tilde{a}\tilde{b}}) - ab. \qquad (15)$$

$\blacksquare$

Remark. In contrast with addition of positive numbers, subtraction of such numbers can lead to large relative errors. For example, if $a = 1 + \varepsilon_*/2$ and $b = 1$, $\widetilde{\tilde{a} - \tilde{b}} = 0$; compared with $a - b = \varepsilon_*/2$, the relative error is 1. This is the gist of why the unshifted FFT-based convolution typically fails in our case.

4.3. Numerical errors in the NC algorithm

Both the NC algorithm and sFFT begin with computing $\tilde{p}_\delta$. Since Hertz and Stormo's method of computing $\tilde{p}_\delta$ introduces another type of lattice errors, we chose to replace it here. The modified NC algorithm (which we still call NC) computes $\tilde{p}_\delta$ by simply enumerating all possible observed frequency tables as in step 1 of sFFT (see Sections 3.1 and 3.2 for more). Thus, according to Lemmas 2 and 3, the bound on the relative error of each entry, $\tilde{p}_\delta(j\delta)$, is determined by A and by $\alpha(j)$, the number of frequency tables whose entropy lies in $[j\delta, (j-1)\delta)$. More precisely, there exists a small, universal constant, c_0 such that with $c_\delta = c_0 A \max_j \alpha(j)$,

$$|\tilde{p}_\delta(s) - p_\delta(s)| \le c_\delta \varepsilon_* |p_\delta(s)|. \qquad (16)$$

In practice, one can increase the size of the lattice M so that c_δ will be sufficiently small.

In the second stage of the NC algorithm, $\tilde{p}_\delta{}^{*L}$ is computed by a naive convolution of $\tilde{p}_\delta$. At each of the $L - 1$ convolutions and for each entry of the convoluted vector, we essentially need to add M products of pairs of positive numbers. By Lemmas 2 and 3, the relative error of each entry is thus increased at every step by $(1 + c_\delta + M)\varepsilon_*$ so that overall

$$|\widetilde{p_\delta^{*L}}(s) - p_\delta^{*L}(s)| \le (M + c_\delta)L\varepsilon_* p_\delta^{*L}(s).$$

Since typically $ML\varepsilon_* \ll 1$, this establishes the reliability of the NC algorithm.

4.4. Error analysis of sFFT

We begin with assessing the quality of our approximation of the DFT operators D and D^{-1} (see Kaneko and Liu [1970] for somewhat similar results). As customary, let $\|x\|_1 = \sum_j |x(j)|$ and let $\|x\|_\infty = \max_j |x(j)|$. Throughout this discussion, we consider $N = 2^K$ as it greatly simplifies the exposition.

Lemma 4. *If* $N = 2^K$ *and* $x = \widetilde{x}$ *then*

$$\|(D_N - \widetilde{D_N})x\|_\infty \le 5K\varepsilon_*\|x\|_1 + 13K^2\varepsilon_*^2\|x\|_1, \tag{17}$$

$$\|(D_N^{-1} - \widetilde{D_N^{-1}})y\|_\infty \le \frac{5K\varepsilon_*}{N}\|y\|_1 + \frac{13K^2\varepsilon_*^2}{N}\|y\|_1. \tag{18}$$

Remark. For our applications, we can safely assume that $K\varepsilon_* \ll 1$. Therefore, when applying (17) and (18), we will absorb the quadratic term in ε_* in the linear term.

Proof. We prove only (17) since the proof of (18) is essentially identical. The proof is by induction on K, where $N = 2^K$, and it is based on the following recursive formula for computing the DFT via FFT (Press *et al.*, 1992): let $\zeta = e^{2\pi i/N}$ and $x \in \mathbb{C}^N$; then

$$(D_N x)(k) = \left(D_{N/2}x_e\right)(k) + \zeta^k \left(D_{N/2}x_o\right)(k), \tag{19}$$

where $x_e, x_o \in \mathbb{C}^{N/2}$ are, respectively, the even and odd entries of x, and k is taken modulo $N/2$ on the right-hand side.

Note that (17) is trivial for $K = 0$, so we need only to establish the inductive step. For $\alpha \in \mathbb{C}$, as before, we denote by $\widetilde{\alpha}$ its machine estimator and define $e_\alpha = \widetilde{\alpha} - \alpha$. For $\alpha, \beta \in \mathbb{C}$, we define

$$e_{\alpha+\beta} = \widetilde{\widetilde{\alpha} + \widetilde{\beta}} - (\alpha + \beta),$$

and similarly for $e_{\alpha\beta}$. It follows from (13) that

$$|e_{\alpha+\beta}| \le (|e_\alpha| + |e_\beta|)(1 + \varepsilon_*) + \varepsilon_*|\alpha + \beta|. \tag{20}$$

Similarly, for $w, z \in \mathbb{C}$ with $|w| = 1$ and assuming $|e_w| \le \varepsilon_*$,[3]

$$|e_{wz}| \le |e_z|(1 + 4\varepsilon_* + 6\varepsilon_*^2) + |z|(4\varepsilon_* + 6\varepsilon_*^2). \tag{21}$$

The latter inequality follows from (12): when applied to $x, y \in \mathbb{C}$ with $x = \widetilde{x}$ and $y = \widetilde{y}$, it holds with $|\varepsilon_{xy}| < 4\varepsilon_* + 2\varepsilon_*^2$. Plugging it into (15) yields the desired result.

Let $\alpha = \left(D_{N/2}x_e\right)(k)$, $z = \left(D_{N/2}x_o\right)(k)$, $w = \zeta^k$, and $\beta = wz$. By induction,

$$|e_\alpha| \le 5K\varepsilon_*\|x_e\|_1 + 13K^2\varepsilon_*^2\|x_e\|_1, \qquad |e_z| \le 5K\varepsilon_*\|x_o\|_1 + 13K^2\varepsilon_*^2\|x_o\|_1.$$

Thus, using (21) and the fact that $|z| \le \|x_o\|_1$, we find that

$$|e_\beta| \le \left[(5K + 4)\varepsilon_* + (13K^2 + 20K + 6)\varepsilon_*^2 + \gamma_3\varepsilon_*^3 + \gamma_4\varepsilon_*^4\right]\|x_o\|_1,$$

where γ_3 and γ_4 are fixed quadratic polynomials in K. Thus,

$$|e_\alpha| + |e_\beta| \le \left[(5K + 4)\varepsilon_* + (13K^2 + 20K + 6)\varepsilon_*^2 + \gamma_3\varepsilon_*^3 + \gamma_4\varepsilon_*^4\right]\|x\|_1,$$

and it follows from (19) and (20) and from $|\alpha + \beta| \le \|x\|_1$ that

$$\begin{aligned} \left|\left[\left(\widetilde{D_N} - D_N\right)(x)\right](k)\right| &\le \left[5(K + 1)\varepsilon_* + (13K^2 + 25K + 10)\varepsilon_*^2 + \gamma_3'\varepsilon_*^3 + \gamma_4'\varepsilon_*^4 + \gamma_5'\varepsilon_*^5\right]\|x\|_1 \\ &\le \left[5(K + 1)\varepsilon_* + 13(K + 1)^2\varepsilon_*^2\right]\|x\|_1, \end{aligned}$$

where the last inequality holds for any usable K (recall that $K = \log_2 N$). This completes the inductive step in proving (17). ■

[3] This assumption clearly depends on the particular implementation of the FFT. We can, however, assume that such an implementation is possible.

Next, we estimate the error in an FFT-based convolution. For the sake of notation sanity, we assume $N = 2^K = ML$.

Lemma 5. *Let $\boldsymbol{q} \in \mathbb{R}^M$, $q_i \geq 0$ with $\|\boldsymbol{q}\|_1 = 1$ (i.e., $\boldsymbol{q}$ is a pmf). Assume that $\widetilde{\boldsymbol{q}} \in \mathbb{R}^M$ satisfies $|q_i - \widetilde{q}_i| < q_i c_\delta \varepsilon_*$. Assume further that $\widetilde{\boldsymbol{q}^{*L}}$ is computed as described in Section 2. Then,*

$$\|\widetilde{\boldsymbol{q}^{*L}} - \boldsymbol{q}^{*L}\|_\infty \leq c(c_\delta + K)L\varepsilon_*,$$

where c is a small universal constant.

Proof. We identify $\boldsymbol{q}$ with a vector in $\mathbb{C}^N$ by padding it with zeros. Note that for $x \in \mathbb{C}^N$, $\|Dx\|_\infty \leq \|x\|_1$ and that $|\|\widetilde{\boldsymbol{q}}\|_1 - 1| < c_\delta \varepsilon_*$. Thus, it follows from Lemma 4 and the triangle inequality that

$$\begin{aligned}\|\widetilde{D}\widetilde{\boldsymbol{q}} - D\boldsymbol{q}\|_\infty &\leq \|D(\widetilde{\boldsymbol{q}} - \boldsymbol{q})\|_\infty + \|(D - \widetilde{D})\widetilde{\boldsymbol{q}}\|_\infty \\ &\leq c_\delta \varepsilon_* \|\boldsymbol{q}\|_1 + cK\varepsilon_* \|\widetilde{\boldsymbol{q}}\|_1 \\ &\leq c(c_\delta + K)\varepsilon_*.\end{aligned} \tag{22}$$

Note that c here means a small constant though it might not be the same constant throughout. Note also that in the spirit of the remark following Lemma 4 we absorb higher powers of ε_* in the linear term.

Define $\boldsymbol{y} \in \mathbb{C}^N$ as $y(i) = \left[(D\boldsymbol{q})(i)\right]^L$ (with $\widetilde{\boldsymbol{y}}$ its analogously defined estimator). From $\|D\boldsymbol{q}\|_\infty \leq 1$ and (22),

$$\|\boldsymbol{y} - \widetilde{\boldsymbol{y}}\|_\infty \leq c(c_\delta + K)L\varepsilon_*. \tag{23}$$

Note that, typically, for most entries $|D\boldsymbol{q}(i)| \ll 1/L$ so $|y(i) - \widetilde{y}(i)| \leq c(c_\delta + K)\varepsilon_*$, but in any case L is relatively small for our applications. Since $\|\boldsymbol{y}\|_1 \leq N$, it follows from (23) that

$$\|\widetilde{\boldsymbol{y}}\|_1 \leq N[1 + c(c_\delta + K)L\varepsilon_*]. \tag{24}$$

At the last step we compute $\widetilde{\boldsymbol{q}^{*L}} = \widetilde{D^{-1}}\widetilde{\boldsymbol{y}}$. Since for $x \in \mathbb{C}^N$ $\|D^{-1}x\|_\infty \leq \frac{1}{N}\|x\|_1$, by using Lemma 4, (23), and (24), we find that

$$\begin{aligned}\|\boldsymbol{q}^{*L} - \widetilde{\boldsymbol{q}^{*L}}\|_\infty &\leq \|D^{-1}(\boldsymbol{y} - \widetilde{\boldsymbol{y}})\|_\infty + \|(D^{-1} - \widetilde{D^{-1}})\widetilde{\boldsymbol{y}}\|_\infty \\ &\leq \frac{1}{N}\|\boldsymbol{y} - \widetilde{\boldsymbol{y}}\|_1 + \frac{cK\varepsilon_*}{N}\|\widetilde{\boldsymbol{y}}\|_1 \\ &\leq c(c_\delta + K)L\varepsilon_*.\end{aligned}$$

■

We are now ready to find what is the overall error of sFFT(s, θ) in estimating $\bar{F}_\delta^{*L}(s)$. Recall that invoking sFFT(s, θ) means θ is given rather than being optimized.

Claim 2. *Let*

$$EB(s, \theta) := c(c_\delta + K)L\Big(\sum_{s' \geq s} e^{-\theta s' + L \log M(\theta)}\Big)\varepsilon_*. \tag{25}$$

Then

$$|\bar{F}_\delta^{*L}(s) - sFFT(s, \theta)| \leq EB(s, \theta). \tag{26}$$

Remarks.

- Note that (26) holds for every θ (and not only the one that minimizes (6)).
- Comparing this bound with the computed p-value, sFFT(s, θ), we can gauge the latter's accuracy. There are, however, more ad hoc ways to assess the accuracy of the p-value as well. For example, the imaginary

part of the p-value should, of course, be 0. Thus, its actual computed value yields an estimate of the level of numerical noise. Alternatively, one can slightly vary θ and observe its effect on the computed p-value.

- Refer to Section 5 for concrete examples of the utility of this upper bound.

Proof. Our proof relies on examination of the accumulated round-off errors at each step of the algorithm. Let $\widetilde{p_\theta}(s) = p_\delta(s)\widetilde{e^{\theta s}}/M_\delta(\theta)$, where $M_\delta(\theta)$ is the moment generating function of p_δ and $\widetilde{p_\theta}$ is short for $\widetilde{p_{\theta,\delta}}$. Note that the error in estimating $M_\delta(\theta)$ can be rather large since $M_\delta(\theta)$ itself can be very large. Fortunately though, we can ignore this error since $M_\delta(\theta)$ is simply a multiplicative factor whose sole utility here is to enhance numerical stability. Thus, we might as well assume that $\widetilde{M_\delta}(\theta) = M_\delta(\theta)$. It therefore follows from (16) that at the end of step 3 of our algorithm (skipping step 2)

$$|\widetilde{p_\theta}(s) - p_\theta(s)| < (c \cdot c_\delta \cdot \varepsilon_*) p_\theta(s).$$

Let $\widetilde{p_\theta^{*L}}$ denote the L-fold convolution computed in step 4 of sFFT. By Lemma 5,

$$\|\widetilde{p_\theta^{*L}} - p_\theta^{*L}\|_\infty \le c(c_\delta + K)L\varepsilon_*. \tag{27}$$

Peeling off the exponential shift, we get the estimate of $p^{*L}(s)$ obtained through the θ shift:

$$\widetilde{p_\delta^{*L}}[\theta](s) = \widetilde{p_\delta^{*L}}(s)\widetilde{e^{-\theta s}}\,[M_\delta(\theta)]^L.$$

It follows from (27) that

$$\|\widetilde{p_\delta^{*L}}[\theta](s) - p_\delta^{*L}\|_\infty \le c(c_\delta + K)Le^{-\theta s + L\log M_\delta(\theta)}\varepsilon_*.$$

The proof is now completed by the definition of $\text{sFFT}(s,\theta) = \sum_{s':s'\ge s} \widetilde{p_\delta^{*L}}[\theta](s')$. ■

4.5. *Which θ?*

Following the definition of $\theta_0 = \theta_0(s_0)$ in (6), we gave an intuitive explanation as to why it should be a good choice of θ. We can now provide additional motivation for that. Following on (25) and (26), we have

$$|\bar{F}_\delta^{*L}(s_0) - \text{sFFT}(s_0)| \le c(c_\delta + K)L\frac{s_{\max} - s_0}{\delta}e^{-\theta s_0 + L\log M(\theta)}\varepsilon_*, \tag{28}$$

where $s_{\max}$ is the maximal possible score. Clearly, the right-hand side is minimized for $\theta = \theta_0$ as defined in (6).

Moreover, a theorem of Bahadur and Ranga Rao (1960) guarantees that as $m \to \infty$

$$P\Big(\frac{1}{m}\sum_{i=1}^{m} I(i) \ge s_0/L\Big) = \frac{e^{-\theta_0 s_0 m/L + m\log M(\theta_0)}}{\sqrt{2\pi m}}(c + o(1)).^4 \tag{29}$$

If we plug $m = L$ in (29), we get the estimate

$$\bar{F}_\delta^{*L}(s_0) \approx c\frac{1}{\sqrt{2\pi L}}e^{-\theta_0 s_0 + L\log M(\theta_0)}.$$

Plugging this estimate into (28), we get the approximate upper bound:

$$|\bar{F}_\delta^{*L}(s_0) - \text{sFFT}(s_0)| \lessapprox \Big[c(c_\delta + K)L\frac{s_{\max} - s_0}{\delta}\sqrt{2\pi L}\varepsilon_*\Big]\bar{F}_\delta^{*L}(s_0).$$

In other words, the *relative error* in our estimation of $\bar{F}_\delta^{*L}(s_0)$ is given by the term in the brackets, which is typically small.

[4] Technically this only holds for s_0/L bigger than the average entropy score of one column, but this condition is satisfied in our applications.

Of course, the last argument hinges on the applicability of the asymptotic behavior described in (29) to a finite $m = L$. While this is typically valid for most s_0, like Hertz and Stormo's large deviation argument, it comes up short for $s_0 \approx s_{\max}$. Nevertheless, the next section provides empirical evidence that $EB(s, \theta_0(s))$, the error bound defined in (25), is usable even for $s_0 \approx s_{\max}$. More important is the implicit statement that sFFT(s) provides an accurate estimate for s all the way up to $s_{\max}$.

5. EMPIRICAL RESULTS

This section presents experimental evidence that supports the claim that sFFT is at times

- significantly more accurate than LD, and
- significantly faster than NC.

In the example that we chose, $n = 20$ (sequences), $M = 10^5$ (lattice size), $A = 4$ (DNA letters), and $q = [0.2499, 0.2501, 0.2497, 0.2503]$ (background frequencies).

5.1. Accuracy

Our benchmark is the NC algorithm whose accuracy we established in Section 4.3. Note, however, that when comparing with Hertz and Stormo's LD algorithm, we need to adjust for the possible lattice errors of NC (Section 4.1).

Let LD(s), NC(s), and sFFT(s) denote the p-values of s as estimated by the respective algorithm. Figure 2 essentially shows $\log_{10} \text{LD}(s)/\text{NC}(s)$, however, to account for possible lattice errors of NC, (8), the graph shown is somewhat more conservative. More precisely, the graph in Fig. 2 is of

$$\psi(s) = \begin{cases} \log_{10} \dfrac{\text{LD}(s)}{LB_\delta(s)} & \text{if } \left|\log_{10} \dfrac{\text{LD}(s)}{LB_\delta(s)}\right| \le \left|\log_{10} \dfrac{\text{LD}(s)}{UB_\delta(s)}\right| \\ \log_{10} \dfrac{\text{LD}(s)}{UB_\delta(s)} & \text{otherwise.} \end{cases} \tag{30}$$

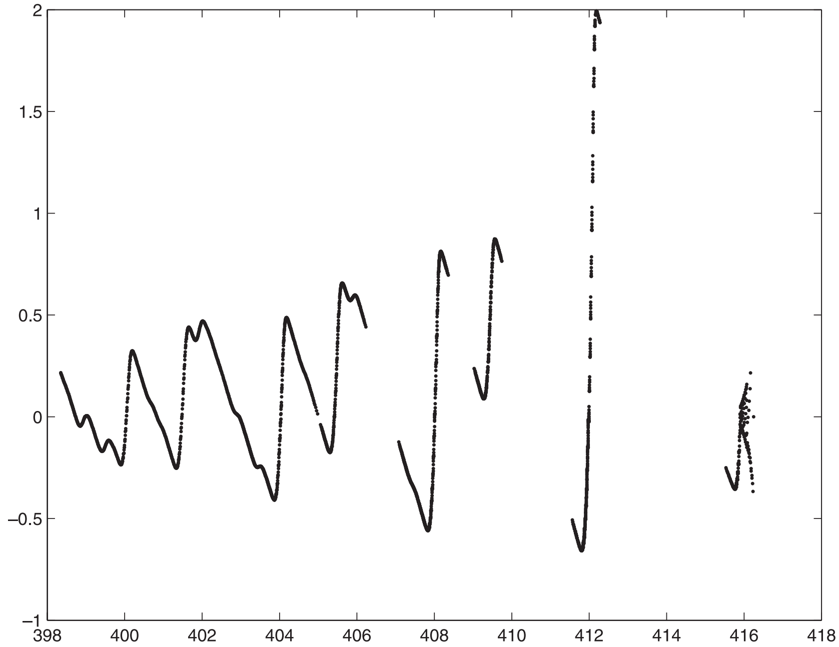

FIG. 2. LD errors. The graph of $\psi(s)$ as defined in (30) is a lower bound on how far off (in $\log_{10}$ units) is $LD(s)$ from the p-value it tries to estimate. The gaps indicate areas of unattainable entropy values.

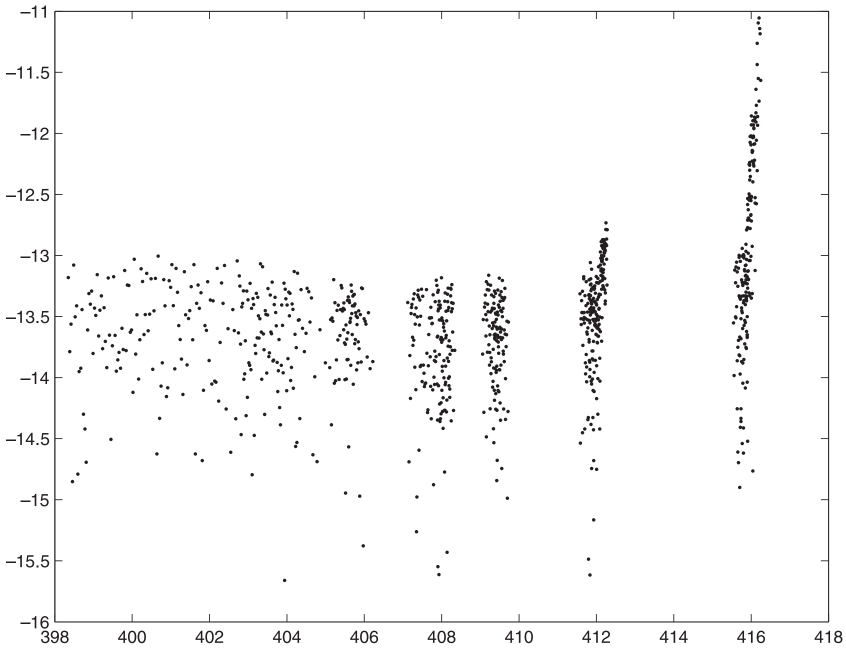

FIG. 3. sFFT errors. This figure illustrates the relative error (in $\log_{10}$ units) between sFFT and NC which is provably accurate.

In contrast with the performance of LD(s) in our example, sFFT(s) stays with NC(s) all the way to $s_{\max}$: Fig. 3 shows the relative error between NC(s) and sFFT(s) for a wide sample $\mathcal{S}$ of values of s near $s_{\max}$ for which $p_\delta^{*L}(s) > 0$. The sample $\mathcal{S}$ was generated with decreasing interspacing as we get closer to $s_{\max}$ so as to get more "resolution" near $s_{\max}$. The gaps in $\mathcal{S}$ are in areas with unattainable values of entropy.

Figure 4 demonstrates the potential utility of the error bound $EB(s, \theta_0)$ specified in (25). The graph shows $\log_{10} EB(s, \theta_0)/\mathrm{NC}(s)$ for s in the previously mentioned sample $\mathcal{S}$, and $\theta_0 = \theta_0(s)$ defined by (6).

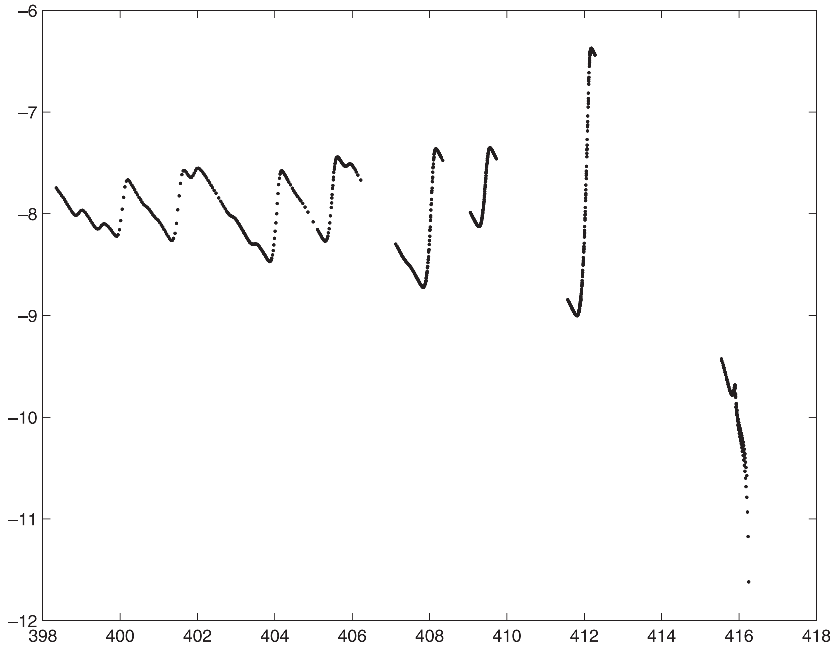

FIG. 4. Sizing $EB(s, \theta_0)$, the bound on the error of sFFT. This figure illustrates the utility of the upper bound given in (26): it depicts $\log_{10} EB(s, \theta_0)/NC(s)$, where $\theta_0 = \theta_0(s)$ is defined by (6).

In studying the utility of $EB(s, \theta_0)$ and the accuracy of sFFT(s), we restricted attention to $s \approx s_{\max}$. This is justified by the discussion in Section 4.5 which ensures us that, away from $s \approx s_{\max}$, $EB(s, \theta_0)$ should be a reasonably tight upper bound and, in particular, that sFFT(s) should be accurate.

5.2. *Speed*

For the example studied here, computing sFFT(s) took 4 seconds for each s while computing NC(s) took 5,215 seconds. Moreover, as mentioned in Section 3.2, sFFT can handle logarithmic computations to avoid underflowed p-values. The associated speed penalty is barely observable. By comparison, a similar extension to NC would be extremely taxing since one would need to carry out essentially $O(L^2M^2)$ calls to “log” and “exp.” Granted, by using tables one can greatly reduce that number of calls but at a price of increased numerical errors and it would still be significantly slower than sFFT.

5.3. *Computing p_δ^{*L} in its entirety*

As mentioned in the introduction, one advantage NC has over LD is that, in fact, it computes the pmf $p_\delta^{*L}(s)$ for all s at the same time. Clearly we cannot hope to achieve this by simply invoking sFFT once, for the same reason that the unshifted FFT typically fails in estimating the p-value. However, calculating a different $\theta_0 = \theta_0(s)$ for every s in the range is clearly an overkill since sFFT typically accurately recovers a range of s values rather than simply the point estimate at s_0. Indeed, with hardly any planning, we could easily find as few as four values of θ with which we were able to recover p_δ^{*L} in its entirety.

More precisely, for each $\theta \in \{-4, 0, 1, 1.5\}$, we generated $\widetilde{p_\delta^{*L}}[\theta]$ using sFFT(θ). Define $u(k) = \left[\widetilde{D}\widetilde{p_{\delta,\theta}}(k)\right]^L$. Selecting only entries $\widetilde{p_\delta^{*L}}[\theta](j\delta)$ for which (see the remarks following Claim 2)

$$\text{Real}\{u(j)\} > 1000\|\text{Imag}\{\boldsymbol{u}\}\|_\infty,$$

we stitched together an estimator $\widetilde{p_\delta^{*L}}$ of p_δ^{*L} which satisfies

$$|\widetilde{p_\delta^{*L}}(j\delta) - p_{NC}(j\delta)| < 4.3 \cdot 10^{-4} p_{NC}(j\delta),$$

where $p_{NC}(j\delta)$ is the reliable estimator of $p_\delta^{*L}(j\delta)$ obtained by NC.

6. CONCLUSION AND FUTURE WORK

We present sFFT, an algorithm which can greatly speed up computing a reliable estimate of the p-value of the entropy score. By applying an exponential shift prior to FFT-based convolution, we are able to retain the accuracy of the rather slow naive convolution ($O(L^2M^2)$) at a speed of FFT-based convolution: $O(LM\log(LM))$.

We anchor the accuracy of sFFT in theoretical and experimental results. One of these allows us to equip sFFT with a guaranty for its accuracy: (26) provides an upper bound on the error which can be compared with the computed p-value.

Our experiments show that by combining a few runs of sFFT with different values of θ, we can potentially recover an estimate of p^{*L} in its entirety. This promising observation requires further study.

Finally, in this paper, sFFT was used to compute the p-value of the entropy score. However, the underlying method may be applicable in a wider context. For example, (26) and the discussion in Section 4.5 hold for any lattice pmf for which (16) holds. More generally, the applicability scope of sFFT requires further study.

ACKNOWLEDGMENTS

We would like to thank Neil Jones and Pavel Pevzner for fruitful discussions on this subject. We would also like to thank the anonymous referee and Niranjan Nagarajan for many insightful comments.

REFERENCES

Bahadur, R.R., and Ranga Rao, R. 1960. On deviations of the sample mean. *Ann. Math. Statist.* 31, 1015–1027.

Bailey, T.L., and Elkan, C. 1994. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. 2nd Int. Conf. on Intelligent Systems for Molecular Biology*, 28–36.

Bailey, T.L., and Gribskov, M. 1998. Combining evidence using p-values: Application to sequence homology searches. *Bioinformatics* 14, 48–54.

Hertz, G.Z., and Stormo, G.D. 1999. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* 15, 563–577.

Jones, N., Keich, U., and Pevzner, P.A. Evaluating profile seeking algorithms. In preparation.

Kaneko, T., and Liu, B. 1970. Accumulation of round-off error in fast fourier transforms. *J. ACM* 17(4), 637–654.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. 1992. *Numerical Recipes in C. The Art of Scientific Computing*, 2nd ed., Cambridge University Press, London.

Rice, J.A. 1995. *Mathematical Statistics and Data Analysis*, 2nd ed., Duxbury Press, Belmont, CA.

Staden, R. 1989. Methods for discovering novel motifs in nucleic acid seqences. *Comput. Appl. Biosci.* 5, 293–298.

Stoer, J., and Bulirsch, R. 1992. *Introduction to Numercal Analysis*, 2nd ed., Springer-Verlag, Berlin.

Stormo, G.D. 2000. DNA binding sites: Representation and discovery. *Bioinformatics* 16(1), 16–23.

Address correspondence to:
Uri Keich
Computer Science Department
Cornell University
4130 Upson Hall
Ithaca, NY 14853

E-mail: keich@cs.cornell.edu