MuST: THE MULTILEVEL SINC TRANSFORM

OREN E. LIVNE * AND ACHI E. BRANDT †

Abstract. A fast multilevel algorithm (MuST) for evaluating an n-sample sinc interpolant at mn points is presented. For uniform grids, its complexity is $25mn\log(1/\delta)$ flops for the sinc kernel and $75mn\log(1/\delta)$ for the sincd kernel, where δ is the target evaluation accuracy. MuST is faster than FFT- and FMM-based evaluations for large n and/or for large δ. It is also applicable to non-uniform grids and to other kernels. Numerical experiments demonstrating the algorithm's practicality are presented.

Key words. Fast multilevel summation algorithm, integral transforms, signal processing, sinc interpolation, oscillatory kernels, Fast Fourier Transform (FFT), linear scaling.

AMS subject classifications. 65D05, 65D15, 65M55

1. Introduction. Interpolation is a fundamental tool that recovers a continuous signal $\mathcal{V}(x)$ from samples $\{\mathcal{U}_j\}_{j\in\mathbb{Z}}$ [14, 26, 28]. The uncertainty principle maintains that either the interpolation window or its spectrum must be infinitely supported [20]. Common schemes ranging from nearest-neighbor and linear interpolation to Hanning and Meijering filters use a finite apodization window [24] at the expense of accuracy loss and aliasing artifacts [28]; conversely, exact recovery of a bandlimited signal from equidistant samples is achieved by the Whittaker-Shannon sinc interpolation formula [26, p. 108],[28]

$$\mathcal{V}(x) = \sum_{j=-\infty}^{\infty} \mathcal{U}_j \operatorname{sinc}\left(\pi\left(\frac{x}{h} - j\right)\right), \qquad \operatorname{sinc}(x) := \frac{\sin(\pi x)}{\pi x}, \tag{1.1}$$

where h is discretization meshsize; it is also the limit of Lagrange polynomial interpolation at infinite window length [11, 29]. (1.1) is nevertheless shunned due to sinc's infinite length, whose truncation cannot be warranted by its slow decay [26, p. 112].

The finite analogue of (1.1) has the discrete convolution form

$$\mathcal{V}(x_k) = \sum_{j=1}^{n} \mathcal{G}(x_k - y_j)\mathcal{U}(y_j), \qquad k = 1, \ldots, mn, \tag{1.2}$$

where $\mathcal{G}(r) := \operatorname{sinc}(r/h)$, $\{y_j := jh\}_{j=1}^{n}$, $h = 1/(n-1)$, $\mathcal{U}(y_j) := \mathcal{U}_j$, $\{x_k\}_{k=1}^{mn} \subset \Omega := [0,1]$, and $m \in \mathbb{N}$ is the expansion factor. The approximation error is $O(e^{-Cn/\log n})$ if $\mathcal{V}$ is smooth [21, 23]. (1.2) also arises in sinc methods for definite and indefinite integration, and for the solution of initial and boundary-value differential and integral equations whose solutions have singularities, infinite domains, or boundary layers [22]. Other kernels of interest are $\operatorname{sincd}(r;h)$ [28] and $\operatorname{sinc}^2(r/h)$ [8]. Indeed, a direct summation of (1.2) requires $O(n^2)$ operations, a prohibitively expensive quixotry leading to the proposition that "optimal reconstruction is possible, but not practical" [11].

*Associate Vice President for Health Sciences IT, University of Utah, 26 South 2000 East Room 5775 HSEB, Salt Lake City, UT 84112, USA. Tel: +1-801-213-3713. Fax: +1-801-581-4297. Email: oren.livne@utah.edu

†Department of Mathematics and Computer Science, The Weizmann Institute of Science, POB 26 Rehovot 76100, Israel. Tel. +972-8-934-3545. Fax: +972-8-934-6023. Email: abrandt@math.ucla.edu

Existing evaluation methods of (1.2) are based on either the Fast Fourier Transform (FFT) or on the Fast Multipole Method (FMM), all requiring $O(mn \log n)$ operations.[1] FFT methods were traditionally developed for equispaced x-grids (e.g. [28]), but can now be applied on non-uniform grids at about twice the cost [1, 8]. They also have the advantage of using standard software packages [16]. FFT evaluations requires a periodic kernel. FMM can also be applied to both uniform and non-uniform grids, but depends on the specific form of the kernel [2]; [16, p.99]. See [3, pp. 344–345] and its bibliography for further literature survey of these methods.

We present an alternative algorithm called Multilevel Sinc Transform (MuST) based on the approach of [4, §5],[12],[13], with two research questions in mind: (a) Is it possible to construct an optimal $O(mn)$ evaluation algorithm of (1.2)? As will be shown in §2, the answer is positive: MuST's complexity is $O(mn \log(1/\delta))$, where δ is the desired evaluation accuracy (defined as the l_1 relative error in $\mathcal{V}$; see Eq. (2.14)). (b) Under what conditions is MuST competitive with FFT/FMM methods? For uniform $\{x_k\}_k$, its hidden constant can be reduced via tabulated local corrections (§2.1.1). MuST becomes faster than FFT when $n \gtrapprox 0.67\delta^{-0.65}$ (§3.2), and is therefore practical only for sufficiently large δ (i.e. low accuracy) and/or large n. For non-uniform grids, the cross-over happens at a much larger n.

This is not a dramatic improvement; FFT and FMM remain as practical for evaluating the sinc transform. Notwithstanding, MuST provides an improvement for uniform grids at moderate desired accuracies. The multilevel approach offers additional benefits because it is based on the kernel's *asymptotic smoothness* (revealed after separating its oscillatory part) rather than on its specific form, periodicity, rate of decay or grid uniformity. It is immediately applicable to a wide range of kernels, and in particular to all aforementioned sinc-variants (§4.1). MuST can also be extended to non-uniform grids in $O(mn \log(1/\delta))$ operations using local refinements or non-uniform coarse grids (§4.2). Finally, while the presented approach is one-dimensional, there exists a (non-trivial) higher-dimensional generalization (see [4, §5.3] and 4.3).

2. Fast Sinc Transform. In this section we consider the case

$$\mathcal{G}(r) = \operatorname{sinc}\left(\frac{r}{h}\right) = \frac{1}{\omega r}\sin(\omega r)\,, \quad \omega := \frac{\pi}{h}\,, \tag{2.1}$$

and as in [28] assume a uniform target grid $x_k := (k+\alpha)h/m$, $k = 1, \ldots, mn$ for some $0 \le \alpha < 1$. Similarly to the "direction splitting" of [4, §5.1], we utilize the identity

$$\sin\left(\omega\left(x_k - y_j\right)\right) = \sin\left(\omega x_k\right)\cos\left(\omega y_j\right) - \cos\left(\omega x_k\right)\sin\left(\omega y_j\right) \tag{2.2}$$

to split (1.2) into three terms – diagonal, even and odd:

[1] Unless a base is specified, $\log x$ refers to the base-10 logarithm of x.

$$\mathcal{V}(x_k) \;=\; v_d(x_k) + \sin(\omega x_k)\, v_e(x_k) - \cos(\omega x_k)\, v_o(x_k)\,, \tag{2.3}$$

$$v_d(x_k) := \sum_{j:|y_j - x_k| \le \eta h} \mathcal{G}(x_k - y_j) u(y_j)\,, \tag{2.4}$$

$$v_e(x_k) := \sum_{j=1}^{n} G(x_k - y_j) u_e(y_j)\,, \quad u_e(y_j) := \cos(\omega y_j)\mathcal{U}(y_j)\,,$$

$$v_o(x_k) := \sum_{j=1}^{n} G(x_k - y_j) u_o(y_j)\,, \quad u_o(y_j) := \sin(\omega y_j)\mathcal{U}(y_j)\,,$$

$$G(r) := \begin{cases} 1/(\omega r)\,, & |r| > \eta h\,, \\ 0\,, & |r| \le \eta h\,. \end{cases} \tag{2.5}$$

Here $\eta = 0.5$ is a small cutoff amount in y-meshsizes around $y = x$ to ensure that $v_{d,e,o}$ all have finite limits as $\alpha \to 0$ (because $\mathrm{sinc}(0) = 1$, even though $1/(\omega r)$ is singular at $r = 0$). $v_d(x_k)$ consists of $O(m)$ terms and is therefore directly evaluated in $O(mn)$ operations. Furthermore, for $y_j = jh$ we have $u_o(y_j) \equiv 0$, hence $v_o(x_k) \equiv 0$. It thus remains to evaluate the middle term, from which we shall omit the e-subscripts to simplify notation. We focus on the fast computation of

$$v(x_k) := \sum_{j=1}^{n} G(x_k - y_j) u(y_j)\,, \qquad k = 1, \ldots, mn \tag{2.6}$$

given the values $u(y_j) := (-1)^j \mathcal{U}(y_j)$, $j = 1, \ldots, n$ using the multilevel algorithm developed in [4, §4],[6],[13]. The separation (2.3) was also utilized by [2], which however chose to evaluate (2.6) using FMM.

2.1. Two-level Evaluation. The algorithm utilizes G's asymptotic smoothness (increasing smoothness as $r \to \infty$) to approximate (2.6) by summations at increasingly coarser levels. Let us first describe a single coarsening step.

Level 1 consists of the original data G, $\mathbf{y} := \{y_j\}_j$, $\mathbf{x} := \{x_k\}_k$ and $u := \{u(y_j)\}_j$. Level 2 is twice-coarser, namely,

$$\mathbf{Y} := \left\{ Y_J := y_0 + JH :\; J = -\frac{p}{2} + 1, \ldots, \lfloor \frac{n-1}{2} \rfloor + \frac{p}{2} \right\}$$

$$\mathbf{X} := \left\{ X_K := x_0 + K\frac{H}{m} :\; K = -\frac{p}{2} + 1, \ldots, \lfloor \frac{mn-1}{2} \rfloor + \frac{p}{2} \right\}$$

where $H = 2h$ and p is some even parameter. $\mathbf{Y}, \mathbf{X}$ are padded to allow a central p^{th}-order polynomial interpolation of coarse-level functions to $\mathbf{y}$ and $\mathbf{x}$, respectively [4] [13, App. A]:

$$f(y_j) = \sum_{J \in \sigma_j} \omega_{jJ} f(Y_J) + C_p f^{(p)}(\xi) H^p\,, \qquad C_p \approx \frac{2}{p!}\left(\frac{p}{2e}\right)^p \tag{2.7a}$$

$$\overline{f}(x_k) = \sum_{K \in \overline{\sigma}_k} \overline{\omega}_{kK} \overline{f}(X_K) + C_p \overline{f}^{(p)}(\overline{\xi}) H^p \tag{2.7b}$$

with weights $\{\omega_{jJ}\}_J, \{\overline{\omega}_{kK}\}_K$; $\xi, \overline{\xi}$ lie in the convex hull of the interpolation stencils

$\sigma_j, \overline{\sigma}_k$, respectively. Applying (2.7a) and (2.7b) to G yields

$$G(x_k - y_j) \approx \tilde{G}(x_k - y_j) := \sum_{K \in \overline{\sigma}_k} \overline{\omega}_{kK} \sum_{J \in \sigma_j} \omega_{jJ} G(X_K - Y_J)\,, \tag{2.8}$$

which is only a good approximation for $|y_j - x_k| \geq O(h)$ due to the singularity at $y_j = x_k$. Using (2.8),

$$\begin{aligned} v(x_k) &= \sum_j \tilde{G}(x_k - y_j) u(y_j) + \left(\sum_j (G - \tilde{G})(x_k - y_j) u(y_j) \right) \\ &= \sum_{K \in \overline{\sigma}_k} \overline{\omega}_{kK} \sum_J G(X_K - Y_J) \sum_{j : J \in \sigma_j} \omega_{jJ} u(y_j) + (v_{\text{local}}(x_k) + e(x_k)) \\ &= \tilde{v}(x_k) + v_{\text{local}}(x_k) + e(x_k)\,, \end{aligned} \tag{2.9}$$

$$\tilde{v}(x_k) := \sum_{K \in \overline{\sigma}_k} \overline{\omega}_{kK} V(X_K) \tag{2.10a}$$

$$V(X_K) := \sum_J G(X_K - Y_J) U(Y_J) \tag{2.10b}$$

$$U(Y_J) := \sum_{j : J \in \sigma_j} \omega_{jJ} u(y_j) \tag{2.10c}$$

$$v_{\text{local}}(x_k) := \sum_{j : |y_j - x| \leq (s+1)H} \left[G(x_k - y_j) - \tilde{G}(x_k - y_j) \right] u(y_j) \tag{2.10d}$$

$$e(x_k) := \sum_{j : |y_j - x| > (s+1)H} \left[G(x_k - y_j) - \tilde{G}(x_k - y_j) \right] u(y_j)\,. \tag{2.10e}$$

These term rearrangements suggest the following algorithm: anterpolate (aggregate) u to $\mathbf{Y}$ using (2.10c), evaluate the coarse-level summation (2.10b), interpolate V to $\mathbf{x}$ via (2.10a) and add the correction (2.10d). The fine-level task (2.6) is thereby reduced to the fast evaluation of coarse-level task (2.10b). Here $s \geq 0$ is the local correction region size in coarse meshsizes; the optimal choice of p and s is discussed in §2.1.3.

2.1.1. Tabulated Corrections. Computing v_{local} comprises a significant portion of the evaluation cost. Superficially, each term requires $O(p)$ operations, totalling to $O(psmn)$ for the entire (2.10d). However, as noted in [4, p. 30], there are only $4(s+1)m+1$ distinct $G - \tilde{G}$ values in all v_{local} sums thanks to $\mathbf{x}$'s and $\mathbf{y}$'s uniformity:

$$\begin{aligned} x_k - y_j &= ((k - jm) + \alpha) \frac{h}{m} \\ v_{\text{local}}(x_k) &= \sum_{j : |k - jm + \alpha| \leq t} d_{k-jm} u(y_j) = \sum_{j : |k - jm| \leq t} d_{k-jm} u(y_j)\,, \\ d_\kappa &:= (G - \tilde{G}) \left((\kappa + \alpha) \frac{h}{m} \right), \quad |\kappa| \leq t \end{aligned} \tag{2.11}$$

with $t := 2(s+1)m$. Once $\{d_\kappa\}_{\kappa=-t}^{t}$ are pre-computed and stored in a table of size $O(sm)$, the complexity of evaluating (2.10d) is reduced to $2smn$ operations (one addition and one multiplication per summand).

2.1.2. Error Estimate. Let $\tilde{v}$ be the result of the two-level algorithm, i.e. $e = \tilde{v} - v$. Substituting the error bound in (2.7a) into (2.8), we obtain

$$\begin{aligned}
\left|(G-\tilde{G})(x_k-y_j)\right| &= \left|G(x_k-y_j) - \sum_{K\in\overline{\sigma}_k} \overline{\omega}_{kK}\left[G(X_K-y_j) + C_p H^p G^{(p)}(x_k-\xi)\right]\right| \\
&= \left|G(x_k-y_j) - \sum_{K\in\overline{\sigma}_k} \overline{\omega}_{kK} G(X_K-y_j) - C_p H^p G^{(p)}(x_k-\xi)\right| \\
&= C_p\left|\left(\frac{H}{m}\right)^p G^{(p)}(\overline{\xi}-y_j) + H^p G^{(p)}(x_k-\xi)\right| \le 2C_p H^m \left|G^{(p)}\left(|x_k-y_j| - \frac{H}{2}\right)\right| \\
&\approx \frac{4}{\omega}\left(\frac{pH}{2e}\right)^p \left(|x_k-y_j| - \frac{H}{2}\right)^{-p-1} \\
&\le 8\left(\frac{pH}{2e}\right)^p \left(|x_k-y_j| - H\right)^{-p} |G(x_k-y_j)| \ .
\end{aligned}$$

Consequently we can bound (2.10e) by

$$\begin{aligned}
|e(x)| &\le \left(\max_{j:|y_j-x|>(s+1)H} \frac{\left|G(x-y_j) - \tilde{G}(x-y_j)\right|}{|G(x-y_j)|}\right) \sum_j |G(x-y_j)|\,|u(y_j)| \\
&\lesssim 8C\left(\frac{p}{2es}\right)^p, \qquad C := \max_x \sum_j |G(x-y_j)|\,|u(y_j)| \ . \qquad (2.12)
\end{aligned}$$

The sinc interpolation formula (1.1) is a conditionally-convergent sum. It is also absolutely convergent under minimal smoothness assumptions on the signal being interpolated, e.g. [14, §IV.A],[25]

$$\sum_j \frac{|\mathcal{U}_j|}{j} < \infty\,, \qquad (2.13)$$

when C is bounded as $n \to \infty$, which will hereafter be assumed unless noted. (2.13) includes all $l^q(\Omega)$ functions for $1 < q < \infty$, and may be relaxed to include stochastic processes that satisfy the Wiener-Khinchin-Einstein Theorem conditions [10, p. 390]. Even if we assume that $\mathcal{U} \in l^\infty(\Omega)$, C increases as $\sum_j h/(x-y_j) = O(\log n)$ at worst, e.g. when $\mathcal{U}_j = (-1)^j \Longleftrightarrow u(y_j) \equiv 1$.

Although it is possible that $\|\mathcal{V}\|_\infty \ll \|\mathcal{U}\|_\infty$ in special cases (for example, $\mathcal{U}_j = (-1)^j$, $m = 1$ and set x_k to the root of $\mathcal{V}(x)$ lying between kh and $(k+1)h$; or a similar setup so that $\{x_k\}_k$ are equidistant), $\|\mathcal{V}\|_\infty$ is typically comparable with $\|\mathcal{U}\|_\infty$. Therefore we define the relative evaluation error as

$$\varepsilon := \frac{\|\tilde{v}-v\|_1}{\|\mathcal{U}\|_1}, \qquad \varepsilon \lesssim 8C\left(\frac{p}{2es}\right)^p . \qquad (2.14)$$

This error definition can easily be replaced by a user-input norm to serve the needs of different applications.

2.1.3. Parameter Optimization. The optimal p and s minimize the computational work for a fixed relative evaluation error δ. The total cost of anterpolation, interpolation and correction is $W = O(pn + pmn + smn) = O((p+s)mn)$; denote by

W_c the coarse-level summation's cost that is constant in p and s. Omitting constants, the constrained minimization problem becomes

$$\begin{cases} W & \propto (p + As)\, mn + W_c \longrightarrow \min \\ \varepsilon & \propto (p/(2es))^p \le \delta \end{cases} \tag{2.15}$$

where A is implementation-dependent. This problem was already solved in [13, §2.4.1]; the optimal values scale as

$$p_{\text{opt}} = p_{\text{opt}}(\delta) = K_1 \log \frac{1}{\delta}, \quad s_{\text{opt}} = s_{\text{opt}}(\delta) = K_2 \log \frac{1}{\delta} \tag{2.16}$$

for sufficiently small δ and large mn. In our implementation, $K_1 \approx 1.1, K_2 \approx 1.1$ (see Fig. 3.1). Using these values, the computational work of the two-level algorithm is

$$W \propto mn\,(K_1 + AK_2) \log \frac{1}{\delta} + W_c =: Kmn \log \frac{1}{\delta} + W_c. \tag{2.17}$$

2.2. Multilevel Evaluation. Directly evaluating (2.10b) is still expensive – about $O(mn^2/4)$ operations. Instead, the two-level algorithm is invoked recursively: U is anterpolated to $\mathbf{y}^3$, followed by a summation at the next-coarser Level 3, an interpolation of the result v^3 back to $\mathbf{x}^2$ and a local correction at Level 2. Denote by $\mathbf{y}^l, \mathbf{x}^l, u^l, v^l$ the data structures at a general level l. The recursion is repeated until Level $L = \lfloor \log_2 n/2 \rfloor$ is reached, whose grids sizes are $O(\sqrt{n})$ and $O(m\sqrt{n})$; v^L is directly summed in $O(mn)$ operations. Algorithm 1 summarizes the entire multilevel procedure; the main call to evaluate (2.6) to accuracy δ is MultilevelEvaluation$(1, \delta, u)$.

Algorithm 1 v^l = MultilevelEvaluation(l, δ, u^l)

if $l = L$ **then**
  Directly sum $\{v^l(x_k)\}_k$ – Eq. (2.10b)
else
  $c \leftarrow l + 1,\ p \leftarrow p_{\text{opt}}(\frac{\delta}{2}), s \leftarrow s_{\text{opt}}(\frac{\delta}{2})$ – Eq. (2.16)
  If table not yet stored for these (p, s), calculate and store (2.11)
  $u^c \leftarrow$ Anterpolation(p, u^l) – Eq. (2.10c)
  $v^c \leftarrow$ MultilevelEvaluation$(c, \frac{\delta}{2}, u^c)$
  $v^l \leftarrow$ Interpolation(p, v^c) – Eq. (2.10a)
  Compute v^l_{local} – Eq. (2.10d); $v^l \leftarrow v^l + v^l_{\text{local}}$
end if

Different coarsening steps may employ different p,s values. Let p_l be the order of anterpolation/interpolation from Level $l+1$ to Level l, and s_l the corresponding local correction size, $l = 1, \ldots, L-1$. The total work of evaluating (2.6) is (see (2.17))

$$W = \sum_{l=1}^{L-1} \left(m2^{-l-1}\right) \left(n2^{-l-1}\right) (p_l + As_l) = mn \sum_{l=1}^{L-1} 4^{-l-1} (p_l + As_l) \tag{2.18}$$

and the total evaluation error satisfies (Eq. (2.14)

$$\varepsilon \lesssim 8C \sum_{l=1}^{L-1} \left(\frac{p_l}{2es_l}\right)^{p_l}. \tag{2.19}$$

If $p_l = p_{\text{opt}}(\delta)$ and $s_l = s_{\text{opt}}(\delta)$ for all l, $\varepsilon = (L-1)\delta$. Instead, we use in Algorithm 1

$$p_l = p_{\text{opt}}(2^{-l}\delta), \quad s_l = p_{\text{opt}}(2^{-l}\delta), \qquad l = 1, \ldots, L-1 \tag{2.20}$$

so that

$$\varepsilon = \sum_{l=1}^{L-1} 2^{-l}\delta < \delta$$

and

$$W = Kmn \sum_{l=1}^{L-1} 4^{-l-1} \log\left(\frac{2^{-l}}{\delta}\right) < \frac{4}{3} Kmn \left(\log\frac{1}{\delta} + \frac{4}{3}\log 2\right). \tag{2.21}$$

A separate local correction table (2.11) is stored for each $l = 1, \ldots, L-1$; a table need only be calculated once per each (p_l, s_l) combination, and then cached for all subsequent invocations of MultilevelEvaluation.

2.3. The MuST Algorithm. Putting it all together, the Multilevel Sinc Transform (MuST) algorithm transforms the data $\{\mathcal{U}(y_j)\}_j$ into $\{u(y_j)\}_j$, calls Algorithm 1 to compute v, directly evaluates the diagonal term v_d, and finally merges the two to recover $\{\mathcal{V}(x_k)\}_k$ using (2.3). If necessary, v and v_d can be computed in parallel, yet the gain is small because v comprises the majority of computing time. Because a relative error ε in v causes the same size error in $\mathcal{V}$, the desired evaluation accuracy δ in $\mathcal{V}$ is passed to MultilevelEvaluation without change. Also note that C and thenceforth the error bound (2.14) is independent of η (for sufficiently large s), therefore $\eta = 0.5$ minimizes the diagonal term work (2.4) with no error increase. Both the diagonal term and the coarsest-level direct summations were efficiently implemented using the tabulation method of §2.1.1.

Algorithm 2 $\mathcal{V} = \text{MuST}(\delta, \mathcal{U})$

1: Directly sum v_d using (2.4)
2: $u(y_j) \leftarrow (-1)^j \mathcal{U}(y_j)\,, j = 1, \ldots, n$
3: $v \leftarrow \text{MultilevelEvaluation}(1, \delta, u)$
4: $\mathcal{V}(x_k) \leftarrow v_d(x_k) + \sin(x_k/h) v(x_k)\,, k = 1, \ldots, mn$ – Eq. (2.3)

Algorithm 2's cost is dominated by Step 3, thus $W = O(mn\log(1/\delta))$ as long as C is bounded. In the extreme case of $C = O(\log n)$, ε is increased by a factor of $O(\log n)$, requiring p and s to be $O(\log(1/\delta) + \log\log n)$; this easily follows by substituting $\delta/\log n$ for δ in (2.15). While this makes the work theoretically bounded by $O(mn(\log(1/\delta) + \log\log n))$, the $\log\log n$ term can be neglected unless n is huge (e.g. $n \geq 10^{10^8}$ for $\delta = 10^{-8}$). See also Table 3.3.

3. Results.

3.1. Parameter Optimization. Algorithm 2 was implemented in Object-Oriented MATLAB 7.9.0. To obtain a machine-agnostic work estimate, W was computed in all experiments using manual flop count facilitated by the Lightspeed library [15].

We first sought to verify (2.16) and determine the values of $K_{1,2}$. Because (2.16) is independent of n, m and u, it is sufficient to carry out a brute-force minimization of (2.15) once for several small n values, for which it is also computationally feasible.

Thus the two-level algorithm was run with fixed n and $m = 1$ for all even $p = 2, \ldots, 14$ and all $s = 1, \ldots, 64$. The work $W(p, s)$ and evaluation error $\varepsilon(p, s)$ (Eq. (2.14)) were tabulated vs. p and s. $\varepsilon(p, s)$ was averaged over 5 experiments with $u(y) = \text{rand}[-1, 1]$.

Due to the discreteness of p and s, the optimal parameters were expected to fluctuate around the continuous approximation (2.16). These fluctuations were damped by regularization. Namely,

$$(p_{\text{opt}}(\delta), s_{\text{opt}}(\delta)) = \underset{\varepsilon(p,s)\le\delta}{\text{argmin}} \left[W(p, s) + \alpha_1 p + \alpha_2 s\right] ,$$

were computed for different δ values and stored. The values $\alpha_1 = 0.2$, $\alpha_2 = 0.6$ were experimentally chosen to achieve the best tradeoff between the increase in $W(p, s)$ and p_{opt}'s and s_{opt}'s smoothness. Figs. 3.1(a-b) depict $p_{\text{opt}}(\delta)/\log(1/\delta)$ and $s_{\text{opt}}(\delta)/\log(1/\delta)$, both of which indeed tend to constants as $\delta \to 0$, and quickly converge to a limit as $n \to \infty$. In practice, we refined (2.16) by fitting general linear functions to the plots (RMSE = 0.15) and rounding up:

$$p_{\text{opt}}(\delta) = 2\lceil(1.05 \log \frac{1}{\delta})/2\rceil, \quad s_{\text{opt}}(\delta) = \lceil 1.3 \log \frac{1}{\delta} - 1.7\rceil . \tag{3.1}$$

This choice was used in all multilevel experiments of §3.2. The corresponding two-level work approached $W \approx 34mn \log(1/\delta) + W_c$ for large n (RMSE = 0.1, Figs. 3.1(c-d)). We note that W was not sensitive to the precise choice of p and s.

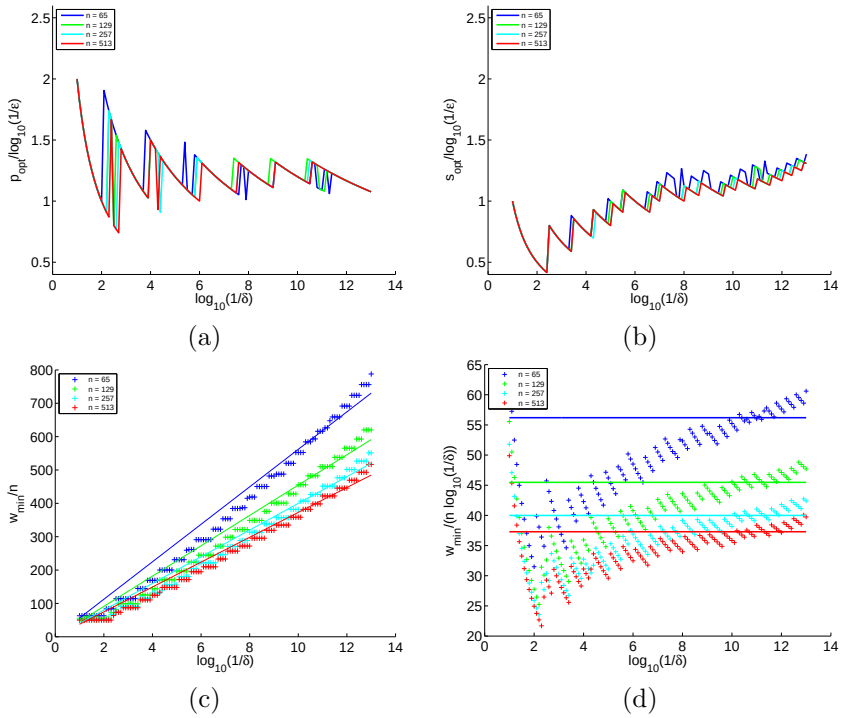

FIG. 3.1. *Two-level optimal parameters and work vs. δ for $n = 129$, 257 and 513. (a) $p_{opt}(\delta)/\log(1/\delta)$; (b) $s_{opt}(\delta)/\log(1/\delta)$; (c) The minimum work $W(p_{opt}(\delta), s_{opt}(\delta))$ (plus symbols) and its linear regression line. (d) $W(p_{opt}(\delta), s_{opt}(\delta))/(mn \log(1/\delta))$ and its average value.*

3.2. Multilevel Complexity. Next, we compared three algorithms of evaluating (1.1) for $\mathcal{U}(y) = \text{rand}[-1, 1]$, $\alpha = 0.2$ and $m = 2$:

(i) Direct summation whose cost is $\approx 30mn$.

(ii) Yaroslavsky's FFT method [28]. Its total cost is roughly $2m$ FFT's; because it is only applicable when n is a power of 2, padding is required for all other values of n, increasing the total cost to $\approx 12mn \log_2 n \approx 40mn \log n$. (Boyd's FMM method [2] has the same complexity with a larger constant, and is therefore less competitive.)

(iii) MuST with target accuracies $\delta = 10^{-d}$, $d = 1, \ldots, 12$. The coarsest grid in all experiments always had at least 160 y-points.

Fig. 3.2(a) depicts the actual ($\log(1/\varepsilon)$) versus desired ($\log(1/\delta)$) number of significant digits in $\mathcal{V}$ in MuST runs. The graph is at or slightly above the line $\varepsilon = \delta$, i.e. the target accuracy was exactly achieved by the parameter settings (3.1).

Fig. 3.2(b) compares the cost of the different algorithms. The MuST complexity is $\approx 25mn \log(1/\delta)$ for all $\delta \leq 10^{-4}$ – slightly better than the two-level prediction for $m = 1$. The hidden constant is tabulated vs. n and m for $\delta = 10^{-8}$ in Table. 3.1.

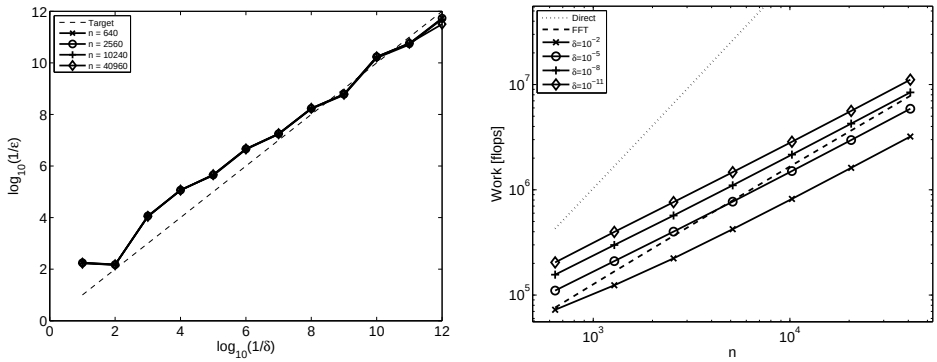

FIG. 3.2. *Multi-level actual accuracy versus target accuracy. (b) Complexity comparison of direct evaluation, FFT and MuST.*

n	$m = 1$	$m = 2$	$m = 4$	$m = 8$
128	37.1	29.4	28.0	27.8
256	42.8	34.3	33.0	33.0
512	39.3	30.8	29.7	29.6
1024	36.3	28.3	27.2	27.1
2048	34.3	26.6	25.6	25.5
4096	33.1	25.7	24.7	24.6

TABLE 3.1
Hidden constant values $W(p_{opt}(\delta), s_{opt}(\delta))/(mn \log(1/\delta))$ versus n and m for $\delta = 10^{-8}$.

MuST quickly becomes faster than direct summation (asymptotically, for all $n \gtrapprox 23 \log(1/\delta)$), as well as faster than FFT when $n \gtrapprox 0.67\delta^{-0.65}$. Table. 3.2 demonstrates different cross-over values. Generally, if the MuST hidden constant is K, the cross-over occurs at $n \gtrapprox 0.67\delta^{-K/40}$; consequently, decreasing K can significantly reduce the cross-over point.

This implies that MuST is superior to its competitors on uniform-grid problems of moderate size or larger, provided that a fixed target accuracy is sought. On the

other hand, in applications where (1.1) arises as a discretized linear system approximating a continuous integral operator, the desired accuracy δ is comparable with the discretization error, which is typically $h^q \sim n^{-q}$ where q is the discretization order. In this case, MuST's complexity is $25qn \log n$ – comparable with FFT, and only marginally better for a first-order discretization.

δ	10^{-2}	10^{-5}	10^{-8}	10^{-11}	10^{-16}
Direct	160	160	160	250	368
FFT	600	2500	40,000	5.3×10^6	7.2×10^9

TABLE 3.2

Cross-over n-values at which MuST becomes faster than its competitors for different target accuracies. Starred values were estimated by extrapolation from Fig. 3.2.

Finally, we compared the MuST performance for smooth and non smooth $\mathcal{U}$ inputs to verify (2.14). Table 3.3 depicts the obtained relative error for a target error of $\delta = 10^{-8}$, increasing problem size and (a) $\mathcal{U}_j = e^{i\beta j}$ with $\beta = 0, \pi/2, \pi$; (b) Runge's function $\mathcal{U}_j = (1 + 25(2y_j - 1)^2)^{-1}$ [18]. The results were practically independent of both $\mathcal{U}$ and n even when u was highly oscillatory ($\beta = \pi$) or when the Runge–Gibbs phenomenon was present. All the results reported in this section remained the same when m was varied, indicating that MuST also scaled linearly with m.

n	$\beta = 0$	$\beta = \pi/2$	$\beta = \pi$	Runge
1280	4.8×10^{-9}	1.07×10^{-8}	5.3×10^{-9}	4.8×10^{-9}
2560	4.8×10^{-9}	1.08×10^{-8}	5.4×10^{-9}	4.7×10^{-9}
5120	4.7×10^{-9}	1.09×10^{-8}	5.3×10^{-9}	4.7×10^{-9}

TABLE 3.3

Actual relative errors for $\delta = 10^{-8}$ and different $\{\mathcal{U}_j\}_j$ choices.

4. Extensions.

4.1. Other Kernels. The MuST algorithm immediately applies to all real- and complex-valued oscillatory kernels of the form

$$\mathcal{G}(r) = G(r)e^{i\omega r}, \qquad G(r) \text{ – asymptotically smooth}\,. \tag{4.1}$$

This includes $\mathcal{G}(r) = \mathrm{sinc}^l(r)$ for all $l > 0$ as well as other kernels arising in acoustics, wave propagation and electromagnetic scattering problems [4, §5],[12].

More generally, $\mathcal{G} = \mathcal{G}(x, y)$ may not be a function of the distance $r = x - y$ and may not have an analytical form; for example, (1.2) may be an image de-noising formula with a spatially-dependent numerical point-spread-function $\mathcal{G}$ whose local behavior is erratic, yet is known (either theoretically or experimentally) to have an asymptotically smooth amplitude $G(x, y)$. MuST can still be applied to this general case, however tabulated local corrections can no longer be used because (2.10d) contains different G-terms for different x_k's. The work consequently rises to $O(mn \log^2(1/\delta))$.

4.1.1. Fast Sincd Transform. Another commonly-encountered sinc variant is the sincd kernel [28, Eq. (2)]

$$\mathcal{G}(r) = \frac{h \sin(\frac{\pi r}{h})}{\sin(\pi r)} = G(r)\sin(\omega r)\,, \quad G(r) := \frac{h}{\sin(\pi r)}\,, \quad \omega := \frac{h}{\pi}. \tag{4.2}$$

This does not fall under the category of (4.1), because G has three singularities at $r_{1,2,3} := -\pi, 0, \pi$ instead of one. We suggest two alternative solutions:

(A) Modify the local correction sum (2.10d) to extend over the region

$$\bigcup_{l=1}^{3} \{j : |y_j - x - r_l| \le (s+1)H\} \ .$$

The local correction tables accordingly become three times larger.

(B) Break the kernel into three kernels with a single singularity each using a partial fraction decomposition. Namely, rewrite

$$G(r) = \frac{\phi(r)}{(r+1)r(r-1)} = \phi(r)\left(\frac{0.5}{r+1} - \frac{1}{r} + \frac{0.5}{r-1}\right) , \tag{4.3}$$

where $\phi(r) := h(r+1)r(r-1)/\sin(\pi r)$ is smooth (Fig. 4.1). Each of the three summands has the form $\phi(r)/(r-r_l)$, hence MuST can be applied to it by modifying the local corrections to extend over $\{j : |y_j - x - r_l| \le (s+1)H\}$.

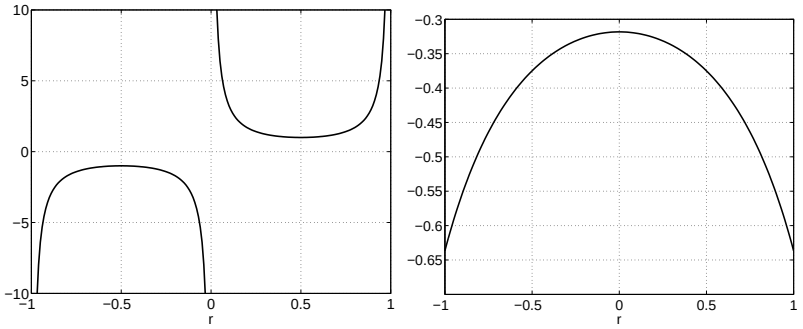

FIG. 4.1. *Left:* $1/\sin(\pi r)$. *Right:* $\phi(r)$ *for* $h = 1$.

(4.3) contains simple singularities only, therefore the Heaviside Cover-Up Method Case 1 [19] can be applied to form the decomposition. This can be generalized to q algebraic singularities with multiplicities $\{\nu_l\}_{l=1}^{q}$ using the general decomposition [27]

$$G(r) = \frac{\phi(r)}{\prod_{l=1}^{q}(r-r_l)^{\nu_l}} = \phi(r)\sum_{l=1}^{\tilde{q}} \frac{A_l}{(r-r_l)^{\tilde{\nu}_l}}$$

for some $\tilde{q}, \{\tilde{\nu}_l\}_l$. Similar decompositions can be developed for other singularity types. We recommend using method A for sincd, which is faster than B because the anterpolation and interpolation steps are not repeated three times. The cost is therefore less than three sinc transforms, or $75mn\log(1/\delta)$ flops. On the other hand, Method B allows using separate coarse grids for different terms in the decomposition, which may prove more flexible for other kernels.

4.2. Non-uniform Grids. First, consider the case where $\mathbf{y}$ and $\mathbf{x}$ are non-uniform grids with a *uniform density* γ, i.e. it is possible to place a uniform grid $\mathbf{Y} = \{Y_J\}_J$ with meshsize $H = O(\gamma)$ over Ω so that there lies a uniformly bounded number ($O(H/\gamma)$) of y_j's within grid cell $[Y_J, Y_{J+1}]$. Similarly define $\mathbf{X}$ over $\mathbf{x}$. Define Level 2 as the grids $\mathbf{Y}$ and $\mathbf{X}$, and subsequent coarsening as in §2. The only change to Algorithm 1 is at the finest level: the anterpolation from $\mathbf{y}$ to $\mathbf{Y}$ and the interpolation from $\mathbf{x}$ to $\mathbf{X}$ require spatially-varying interpolation weights, each of which can be

computed using barycentric interpolation [9] in $11mn\log(1/\delta)$. Furthermore, v_o no longer vanishes, thus Algorithm 2 invokes Algorithm 1 *twice* to compute (2.3).

If $\mathbf{y}$'s density (similarly $\mathbf{x}$'s) varies over the domain, e.g. when using Chebyshev nodes to minimize the Runge-Gibbs phenomenon in the interpolation (1.1), multilevel efficiency is maintained by introducing *local refinements* [7],[13, §4.1]: starting from the global uniform grid $\mathbf{Y}$, finer "patches" are introduced only over the areas of locally-high density. Patches are recursively nested until there are only a bounded number of y_j's within a cell, organized in a multilevel patch tree. Importantly, all patches are uniform grids to keep inter-level interpolations simple and efficient. Here tabulated local corrections should be replaced with the sliding-window technique described in [13]; the overall work should then amount to about $85mn\log(1/\delta)$ flops.

An simpler alternative approach is to use a non-uniform coarse grids ($\mathbf{X} = \{x_{2k}\}_{k=1}^{[mn/2]}$, $\mathbf{Y} = \{y_{2k}\}_{k=1}^{[n/2]}$, appropriately padded as explained in §2). A different error analysis would be required here, as the local correction region size varies over the domain.

4.3. Higher Dimensions. Whilst this paper presents a one-dimensional algorithm, it can be generalized to higher dimensions. An $O(mn\log n)$ multi-level evaluation of integral transforms with oscillatory kernels in d dimensions, $d \geq 2$ was first described in [4, §5] and subsequently implemented for $d = 2$ in [17]. Instead of the directional separation (2.3), the general algorithm separates directions at *all* levels, where the number of directions increases on increasingly coarser levels.

The 1-D MuST can be applied along one dimension at a time to evaluating the separable d-dimensional sinc and sinc^2 kernels considered by [8]. Note that the method of [8] cannot be applied to the full 2-D sinc kernel $\mathrm{sinc}(\|x-y\|_2)$, because the analytical form of the kernel's Fourier transform no longer holds (Eq. (8) and the first equation in §2.2 therein).

4.4. Other Applications. The multilevel evaluation can be efficiently parallelized to multiple processors. Anterpolation, interpolation and local corrections at each level require $O(\log(1/\delta))$ operations per data point, each of which can be independently processed. Because there are $O(\log n)$ levels, the parallel efficiency is $O(\log(1/\delta)\log n)$.

When convenient, the coarsest-level direct summation can be replaced by a faster FFT evaluation. This owes to the flexibility of choosing coarse grids: the coarsest y- and x-grids can always be padded to contain 2^N points at a negligible cost.

The multilevel evaluation can be naturally embedded to compute residuals within an iterative solution of integral equations [5]: the coarse levels are reused by the multilevel solution cycle; δ is matched to the estimated accuracy of the current iterant.

5. Conclusion. A fast multilevel sinc transform (MuST) algorithm was presented. It utilizes the asymptotic smoothness associated with the sinc kernel, therefore its performance is independent of the given samples $\{\mathcal{U}_j\}_j$. With a complexity of $25mn\log(1/\delta)$ for sinc and $75mn\log(1/\delta)$ for sincd, MuST is competitive with FFT- and FMM-based methods on uniform grids for moderate evaluation accuracies, and can be applied to a wider variety of kernels. Numerical experiments were performed in MATLAB ; a next natural step left for the future is an optimized parallel software implementation for integration into practical applications.

6. Acknowledgements. This paper is dedicated to J. Brahms' Piano Concerto No. 2 in B-flat major, Op. 83.

REFERENCES

[1] J. P. Boyd, *A fast algorithm for Chebyshev and Fourier interpolation onto an irregular grid*, J. Comp. Phys., 103 (1992), pp. 243–257.

[2] ———, *Multipole expansions and pseudospectral cardinal functions: A new generalization of the Fast Fourier Transform*, J. Comp. Phys., 102 (1992), pp. 184–186.

[3] ———, *Chebyshev and Fourier Spectral Methods*, Dover, Mineola, New York, second ed., 2000.

[4] A. Brandt, *Multilevel computations of integral transforms and particle interactions with oscillatory kernels*, Computer Physics Communications, 65 (1991), pp. 24–38.

[5] A. Brandt and A. A. Lubrecht, *Multilevel matrix multiplication and fast solution of integral equations*, J. Comput. Phys., 90 (1990), pp. 348–370.

[6] A. Brandt and C. H. Venner, *Multilevel evaluation of integral transforms with asymptotically smooth kernels*, SIAM J. Sci. Comput., 19 (1998), pp. 468–492.

[7] The Carl, F. Gauss, Minerva Center, For Scientific Computation, Achi Brandt, Kees Venner, A. Brandt, and C. H. Venner, *Multilevel evaluation of integral transforms on adaptive grids*, 1996.

[8] L. Greengard, J. Lee, and Souheil Inati, *Fast sinc transform and image reconstruction from nonuniform samples in k-space*, Comm. App. Math. Comp. Sci., 1 (2006), pp. 121–131.

[9] N. J. Higham, *The numerical stability of barycentric lagrange interpolation*, IMA J. Numer. Anal, 24 (2004), pp. 547–556.

[10] K. Iniewski, *Wireless Technologies: Circuits, Systems, and Devices*, CRC Press, 2008.

[11] O. Kreylos, *Sampling theory 101*. `http://idav.ucdavis.edu/~okreylos/PhDStudies/Winter2000/SamplingTheory.html`, 2009.

[12] O. Livne, A. Brandt, and A. Boag, *Multigrid analysis of scattering by large planar structures*, Micro. Opt. Tech. Lett., 32 (2002), pp. 454–458.

[13] O. E. Livne and A. Brandt, *N roots of the secular equation in $O(N)$ operations*, SIAM J. Mat. Anal., 24 (2002), pp. 439–453.

[14] Erik Meijering, *A chronology of interpolation: From ancient astronomy to modern signal and image processing*, in Proceedings of the IEEE, 2002, pp. 319–342.

[15] T. Minka, *The Lightspeed MATLAB toolbox*. `http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/`, 2010. [Online; accessed August 10, 2010].

[16] D. Potts, G. Steidl, and A. Nieslony, *Fast convolution with radial kernels at nonequispaced knots*, Numer. Math., 98 (2004), pp. 329–351.

[17] I. H. Ramirez, *Multilevel Multi-Integration for Acoustics*, PhD thesis, University of Twente, Enschede, The Netherlands, 2005.

[18] C. Runge, *Über empirische funktionen und die interpolation zwischen quidistanten ordinaten*, Zeitschrift fr Mathematik und Physik, 46 (1901), pp. 224–243.

[19] J. Senning, *Heaviside "cover-up" method for partial fractions*. `http://www.math-cs.gordon.edu/courses/ma225/handouts/heavyside.pdf`, 2009. [Online; accessed August 10, 2010].

[20] A. Sitaram, *Encyclopaedia of Mathematics*, Springer, 2002, ch. Uncertainty principle.

[21] F. Stenger, *Numerical methods based on Sinc and analytic functions*, vol. 20 of Springer Series in Computational Mathematics, 1993.

[22] F. Stenger, *Handbook of Sinc Numerical Methods*, Numerical Analysis and Scientific Computation Series, CRC Press, December 2010.

[23] M. Sugihara and T. Matsuo, *Recent developments of the sinc numerical methods*, J. Comput. Appl. Math., 164-165 (2004), pp. 673–689.

[24] P. Thévenaz, T. Blu, and M. Unser, *Interpolation revisited*, IEEE Trans. Med. Imaging, 19 (2000), pp. 739–758.

[25] A. Y. Trynin, *Tests for pointwise and uniform convergence of sinc approximations of continuous functions on aclosed interval*, Sbornik: Mathematics, 198 (2007), p. 1517.

[26] M. N. Wernick and J. N. Aarsvold, *Emission Tomography: The Fundamentals of PET and SPECT*, Academic Press, San Diego, CA, December 2004.

[27] Wikipedia, *Partial fraction*. `http://en.wikipedia.org/wiki/Partial_fraction`, 2010. [Online; accessed August 10, 2010].

[28] L. P. Yaroslavsky, *Signal sinc-interpolation: a fast computer algorithm*, Bioimaging, 4 (1996), pp. 225–231.

[29] M. M. J. Yekta, *Equivalence of the lagrange interpolator for uniformly sampled signals and the scaled binomially windowed shifted sinc function*, Dig. Sig. Proc., 19 (2009), pp. 838–842.