Journal of Computational Physics 228 (2009) 7086–7108

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A geometric nonuniform fast Fourier transform

Ian Sammis, John Strain *

Department of Mathematics, 970 Evans Hall #3840, University of California, Berkeley, CA 94720-3840, United States

ARTICLE INFO

Article history:
Received 18 December 2008
Received in revised form 15 June 2009
Accepted 17 June 2009
Available online 27 June 2009

Keywords:
Fourier transform
B-splines

ABSTRACT

An efficient algorithm is presented for the computation of Fourier coefficients of piecewise-polynomial densities on flat geometric objects in arbitrary dimension and codimension. Applications range from standard nonuniform FFTs of scattered point data, through line and surface potentials in two and three dimensions, to volumetric transforms in three dimensions. Input densities are smoothed with a B-spline kernel, sampled on a uniform grid, and transformed by a standard FFT, and the resulting coefficients are unsmoothed by division. Any specified accuracy can be achieved, and numerical experiments demonstrate the efficiency of the algorithm for a gallery of realistic examples.

1. Introduction

Fourier coefficients

$$\widehat{f}(k) = \int_{-\pi}^{\pi} f(x) e^{\imath k x}\,\mathrm{d}x, \quad |k| \leqslant K$$

constitute an indispensable tool of science. They are accurately approximated by the trapezoidal rule

$$\widetilde{f}(k) = \frac{1}{2N}\left(\frac{1}{2}f(-\pi)e^{-\imath k\pi} + \sum_{-N+1}^{N-1} f(\pi j/N)e^{\imath k\pi j/N} + \frac{1}{2}f(\pi)e^{\imath k\pi}\right)$$

and efficiently evaluated by the FFT [1] when f is a smooth periodic function and $N \geqslant K$. They are equally indispensable but much harder to compute when f is a nonsmooth or nonperiodic function, or a singular distribution. Typically such distributions are defined by a density μ on a m-dimensional submanifold Γ of R^n, according to the recipe

$$\int_{R^n} \varphi(x) f(x)\,\mathrm{d}x = \int_{\Gamma} \varphi(\gamma)\mu(\gamma)\,\mathrm{d}\gamma$$

for all smooth compactly-supported "test functions" φ on R^n. Their Fourier coefficients are defined by

$$\widehat{f}(k) = \int_{R^n} e^{\imath k^T x} f(x)\,\mathrm{d}x = \int_{\Gamma} \mu(\gamma) e^{\imath k^T \gamma}\,\mathrm{d}\gamma$$

for integer wave vectors $k = (k_1, k_2, \ldots, k_n)^T$. The computation of such coefficients is a well-known ingredient of semiconductor mask design [2]. They also occur in classical algorithms such as Ewald summation [3–7], which separates the inverse

* Corresponding author. Tel./fax: +1 510 642 8204.
E-mail address: strain@math.berkeley.edu (J. Strain).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.06.027

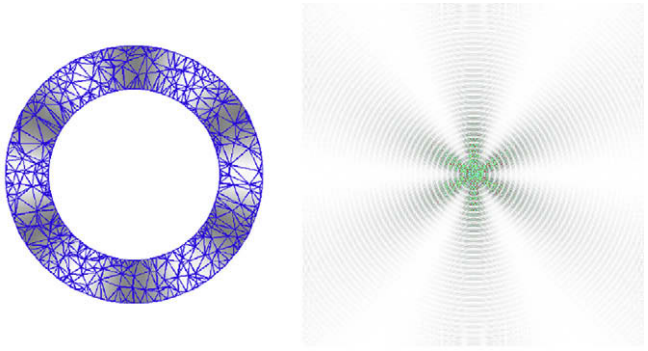


Fig. 1. Polynomials on 760 triangles approximate a sinusoid on an annulus (a), yielding the 128^2 Fourier coefficients with intensity and phase plotted in (b).

$$(-\Delta)^{-1} f(x) = \int_C G(x-y) f(y) dy \tag{1}$$

of an unbounded differential operator into two components

$$G(x) = G_\tau(x) + G_L(x) : \tag{2}$$

a smooth global component

$$G_\tau(x) = \frac{1}{|C|}\left(\tau - \sum_{0 \neq k \in \mathbb{Z}^n} \frac{e^{-\tau|k|^2}}{|k|^2} e^{ik^T x}\right) \tag{3}$$

efficiently represented in the Fourier basis, and a local singular component

$$G_L(x) = -\sum_{k \in \mathbb{Z}^n} \int_0^\tau (4\pi t)^{n/2} e^{-|x-2\pi k|^2/4t} dt \tag{4}$$

efficiently represented in real space, for any $\tau > 0$. We present an efficient algorithm for the computation of accurate Fourier coefficients for a distribution f composed of polynomials μ supported on a set Γ of arbitrary-dimensional simplices – points, line segments, triangles, tetrahedra and so forth (Fig. 1). This geometric nonuniform fast Fourier transform (GNUFFT) delivers performance comparable to specialized nonuniform FFT techniques [8–10] for point data sets, and essentially optimal performance for general geometric data.

The GNUFFT proceeds in three steps. First, the singular distribution f is regularized into a smooth periodic function $f_h = \rho_h^d \star f$ by convolution with a smoothing kernel ρ_h^d. The efficiency of the GNUFFT is ensured by choosing a piecewise-polynomial smoothing kernel with minimal support $O(h)$ and applying it efficiently to the piecewise-polynomial density μ on each simplex in Γ. Second, the Fourier coefficients

$$\widehat{f_h}(k) = \widehat{\rho_h^d}(k)\widehat{f}(k)$$

are accurately approximated by trapezoidal quadrature and efficiently evaluated by the standard FFT. Finally, the smoothing is removed by division in Fourier space. The total error in these steps is analyzed and controlled by balancing the effects of the various smoothing, quadrature and oversampling parameters.

2. Definitions

The GNUFFT operates on the "simplex-polynomial distributions" (SPDs) defined in Section 2.1, which approximate the familiar point sources, line charges, and area charges of potential theory. The multidimensional Bernstein polynomials defined in Section 2.2 provide a stable polynomial basis for SPDs on simplices of arbitrary dimension. The B-spline smoothing kernel described in Section 2.3 permits exact quadrature on simplices. Table 1 summarizes the notation.

2.1. Simplex-polynomial distributions

The GNUFFT transforms the class of piecewise-polynomial densities on simplices defined below.

Definition 1. An m-dimensional simplex in R^n is the convex hull

$$\Delta = \{x = vt = t_0 v_0 + \cdots + t_m v_m | t_j \geqslant 0,\ t_0 + \cdots + t_m = 1\} \tag{5}$$

Table 1
Definitions of symbols.

Symbol	Definition
n	Dimension of ambient space R^n
C	Standard cubical cell $[-\pi,\pi]^n$ in R^n
c	Point of uniform grid with spacing h on C
m	Simplex dimension $m \leqslant n$
Δ	Simplex of dimension m, convex hull of $m+1$ vertices
$\lvert\Delta\rvert$	m-dimensional length, area or volume of simplex
v_j	Simplex vertex $v_j \in R^n$
v	Matrix of vertices $v = (v_0, v_1, \ldots, v_m) \in R^{n\times(m+1)}$
α	Multiindex $(\alpha_0, \ldots, \alpha_m)$ of nonnegative integer indices
$\lvert\alpha\rvert$	Order $\lvert\alpha\rvert = \alpha_0 + \cdots + \alpha_m$ of multiindex α
t	Barycentric coordinates of simplex point $x = vt = t_0 v_0 + \cdots + t_m v_m$
B	Set of barycentric coordinates
$\mathrm{d}t$	Lebesgue measure on simplex
d	Degree of a polynomial
p	Polynomial of n or m variables x or t
B^d_α	Bernstein polynomial of degree d and multiindex α
t_q	Quadrature point
w_q	Quadrature weight
Q	Number of quadrature points and weights
k	Fourier wavenumber with integer entries $(k_1, \ldots, k_n)$
$\lvert k\rvert$	Norm $\lvert k\rvert = \max(\lvert k_1\rvert, \ldots, \lvert k_n\rvert)$ of wavenumber k
$e_k(x)$	Fourier basis function $e^{-ik^T x}$ with wavenumber k on standard cell C
K	Output Fourier wavenumber bound $\lvert k\rvert \leqslant K$
s	Oversampling factor for FFT
ρ	Piecewise-polynomial B-spline smoothing kernel ρ^d_h
h	Width parameter $h = \pi/Ks$ of B-spline smoothing kernel
d	Degree of B-spline smoothing kernel

of a collection of $m+1$ points v_j in R^n, which form the $m+1$ columns of an n by $m+1$ matrix v. A point $x = vt \in \Delta$ has barycentric coordinates $t = (t_0, \ldots, t_m) \in B$, where $B = \{t \in \mathbb{R}^{m+1} | t_j \geqslant 0, t_0 + \cdots + t_m = 1\}$.

Definition 2. A function $\varphi : R^n \to R$ is a test function if φ has continuous derivatives $\partial^\alpha\varphi$ of all orders $|\alpha| \geqslant 0$, and vanishes outside a bounded set. A sequence φ_k of test functions converges to 0 if every partial derivative $\partial^\alpha\varphi_k \to 0$ uniformly on R^n as $k \to \infty$.

Definition 3. The space D' of distributions consists of all continuous real-valued linear functionals f on the space D of test functions.

Example 1. Every piecewise-continuous function f on R^n is a distribution which maps φ to

$$f(\varphi) = \int_{R^n} f(x)\varphi(x)\,\mathrm{d}x. \tag{6}$$

Its distributional derivative $\partial^\alpha f$ maps φ to

$$\partial^\alpha f(\varphi) = (-1)^{|\alpha|} f(\partial\varphi) = \int_{R^n} f(x)(-1)^{|\alpha|}\partial^\alpha\varphi(x)\,\mathrm{d}x \tag{7}$$

for any multiindex α, generalizing the usual integration by parts formula.

Definition 4. A simplex-polynomial distribution (SPD) over R^n is a linear functional $f \in D'$ defined by

$$f(\varphi) := \sum_{j=1}^{N} \int_{\Delta_j} p_j(x)\varphi(x)\,\mathrm{d}x. \tag{8}$$

Here each Δ_j is a m_j-dimensional simplex, each p_j is a multivariate polynomial in the variable $x \in \Delta_j$, and $\mathrm{d}x$ is Lebesgue measure of dimension m_j on each Δ_j. For simplicity an SPD may be written

$$f = \sum_{j=1}^{N} p_j\Delta_j, \tag{9}$$

where each simplex is identified with the distribution which integrates test functions over the simplex. If Δ is a zero-dimensional simplex consisting of a single point v_0, and p is any polynomial, then $f = p\Delta$ is defined by

$$f(\varphi) = \int_{\Delta} \varphi(x)p(x)\mathrm{d}x = \varphi(v_0)p(v_0). \tag{10}$$

Example 2. If every simplex Δ_j is zero-dimensional, so $\Delta_j = \{v_j\}$ for all j, then the SPD f is a distribution of point sources which is often written as a function

$$f(x) := \sum_{j=1}^{N} c_j \delta(x - v_j), \tag{11}$$

where $c_j = p_j(v_j)$ and δ is the usual Dirac delta-function.

Example 3. If every simplex Δ_j is a one-dimensional interval $\Delta_j = [v_{j0}, v_{j1}]$ in ambient dimension $n = 1$, then

$$f(\varphi) = \sum_{j=1}^{N} \int_{v_{j0}}^{v_{j1}} f(x)p_j(x)\,\mathrm{d}x. \tag{12}$$

Thus the simplex-polynomial distribution f can be identified with a piecewise-polynomial function. Since every piecewise-polynomial function with finite N may be written in this form, SPDs may be considered generalizations of piecewise-polynomial functions.

Example 4. Consider a triangular region Δ_1 in the plane R^2 with vertices $(0,0), (L,0)$, and $(0,L)$, supporting an areal charge density

$$p_1(x,y) = C(L^2 - x^2)(L^2 - y^2), \tag{13}$$

where C has units of $(\text{charge})/(\text{length})^6$. Let the boundary line segments Δ_2 through Δ_4 of the triangle support a uniform linear charge density $p_2 = p_3 = p_4 = -\frac{C}{16}L^5$, and a point charge of strength $p_5 = \frac{CL^6}{6}$ rest at the origin $\Delta_5 = (0,0)$. Then the SPD $f = \sum p_j \Delta_j$ maps any test function φ to

$$\begin{aligned} f(\varphi) = &\int_0^L \int_0^{L-x} \varphi(x,y)C(L^2 - x^2)(L^2 - y^2)\,\mathrm{d}y\,\mathrm{d}x - \int_0^L \varphi(0,y)CL^5/16\,\mathrm{d}y - \int_0^L \varphi(x,0)CL^5/16\,\mathrm{d}x \\ &- \sqrt{2}\int_0^L \varphi(s,L-s)CL^5/16\,\mathrm{d}s + \varphi(0,0)CL^6/6. \end{aligned} \tag{14}$$

Example 5. The definition of the SPD distinguishes between degenerate and lower-dimensional simplices. A triangle with vertices at (0,0), (1,0), and $(0,\epsilon)$ supporting the constant density 1 is different from the line segment connecting (0,0) and (1,0), supporting the constant density 1. The former is a two-dimensional simplex of nearly zero total mass, while the latter is a one-dimensional object of total mass 1.

Definition 5. The Fourier coefficients of an SPD

$$f(x) := \sum_{j=1}^{N} p_j(x)\Delta_j(x), \tag{15}$$

with simplices Δ_j inside the standard cube $C = [-\pi,\pi]^n$ are defined for $k \in Z^n$ by

$$\widehat{f}(k) := \sum_{j=1}^{N} \int_{\Delta_j} p_j(x)e_k(x)\,\mathrm{d}x, \tag{16}$$

where $e_k(x)$ is the kth Fourier mode

$$e_k(x) := e^{-\imath k^T x}. \tag{17}$$

2.2. Multidimensional Bernstein polynomials

The polynomials p_j in an SPD can be efficiently represented in barycentric coordinates t on each simplex Δ_j by the standard multidimensional Bernstein basis [11] described below.

Definition 6. The Bernstein polynomials B_j^d of a single variable t are defined by

$$B_j^d(t) := \binom{d}{j}(1-t)^{d-j}t^j = \binom{d}{j}t_0^{d-j}t_1^j, \quad 0 \leqslant j \leqslant d \tag{18}$$

in barycentric coordinates $t = (t_0, t_1) = (1 - t, t)$ on the standard interval $\Delta := [0,1]$.

Bernstein polynomials form a basis B^d for the space of polynomials of degree d, in which any polynomial p is represented by a vector a of coefficients: $p(t) = a^T B^d(t) = \sum_\alpha a_\alpha B^d_\alpha(t)$. They are convenient for integration because

$$\int_0^1 B^d_j(t)\,\mathrm{d}t = \frac{1}{d+1}. \tag{19}$$

Definition 7. For a multiindex $\alpha = (\alpha_0, \ldots, \alpha_m)$ of order $|\alpha| := \alpha_0 + \cdots + \alpha_m = d$, the m-dimensional Bernstein polynomial B^d_α is defined by

$$B^d_\alpha(t) := \binom{d}{\alpha} t^\alpha, \tag{20}$$

where $t^\alpha := t_0^{j_0} \cdots t_m^{j_m}$, and $\binom{d}{\alpha}$ is the usual multinomial coefficient [12]

$$\binom{d}{\alpha} := \frac{d!}{\alpha_0! \cdots \alpha_m!}. \tag{21}$$

Due to the leading 0 index, a Bernstein polynomial on an m-dimensional simplex has $m+1$ indices and depends on $m+1$ barycentric coordinates.

Example 6. The one-dimensional Bernstein polynomials B^d_α are multidimensional Bernstein polynomials with $m = 1$ and $\alpha = (d-j, j)$, so

$$B^d_j(t) = B^d_{(d-j\,j)}(1-t, t). \tag{22}$$

Lemma 1. *For each $d \geqslant 0$, the integral*

$$\int_B B^d_\alpha\,\mathrm{d}t \tag{23}$$

is independent of α.

Proof. It suffices to show that the integral over B of B^d_α is equal to the integral over B of B^d_β, where β precedes α in lexicographic order. Let α be any multiindex other than $(d, 0, \ldots, 0)$. Then for some $k > 0$, a multiindex component $\alpha_k \neq 0$. Then

$$\int_B B^d_\alpha(t)\mathrm{d}t = \binom{d}{\alpha} \int_B t_0^{\alpha_0} \cdots t_k^{\alpha_k} \cdots t_m^{\alpha_m}\mathrm{d}t \tag{24}$$

$$= \binom{d}{\alpha} |B| \int_0^1 \cdots \int_0^{t_0+t_k} t_0^{\alpha_0} \cdots t_k^{\alpha_k} \cdots t_m^{\alpha_m}\mathrm{d}t_1 \cdots \mathrm{d}t_{k-1}\mathrm{d}t_{k+1} \cdots \mathrm{d}t_m\mathrm{d}t_k \tag{25}$$

since the factors dt_j in the differential form may be permuted freely as long as the limits of integration are adjusted accordingly. Integration by parts with respect to t_k moves an exponent from the t_k factor to the $t_0 = 1 - t_1 - \cdots - t_m$ factor, and yields

$$\int_B B^d_\alpha(t)\mathrm{d}t = \int_B \binom{d}{\alpha} \frac{\alpha_k}{\alpha_0 + 1} t_0^{\alpha_0+1} \cdots t_k^{\alpha_k - 1} \cdots t_m^{\alpha_m}\,\mathrm{d}t = \int_B B^d_{\alpha - e_k + e_0}(t)\,\mathrm{d}t, \tag{26}$$

where

$$e_k := (\overbrace{0, \ldots, 0}^{k \text{ zeros}}, 1, \overbrace{0, \ldots, 0}^{m-k \text{ zeros}}) \in Z^{m+1}. \tag{27}$$

The boundary terms vanish because $t_0 = 0$ at one boundary, $t_k = 0$ at the other, and both factors appear with positive exponent in the boundary term. Hence all $\binom{d+m}{m}$ of the polynomials B^d_α have the same integral over the standard simplex B. □

Corollary 1. *Let B be the standard m-dimensional simplex defined by $t_k \geqslant 0, \sum_{k=0}^m t_k = 1$. Then*

$$\int_B B^d_\alpha(t)\mathrm{d}t = \frac{1}{m!\binom{m+d}{m}}, \tag{28}$$

where $\mathrm{d}t$ is m-dimensional Lebesgue measure $\mathrm{d}t = \mathrm{d}t_1 \cdots \mathrm{d}t_m$.

Proof. Since Bernstein polynomials interpolate constants exactly,

$$\sum_{|\alpha|=d} B^d_\alpha = 1. \tag{29}$$

Thus integration gives

$$\sum_{|\alpha|=d} \int_B B_\alpha^d(t)\,\mathrm{d}t = \int_B \mathrm{d}t = \frac{1}{m!}. \tag{30}$$

All $\binom{m+d}{m}$ terms of the sum are identical by Lemma 1, with common value

$$\int_B B_\alpha^d(t)\,\mathrm{d}t = \frac{1}{m!\binom{d+m}{m}}. \qquad \square \tag{31}$$

Corollary 2. *A polynomial $p(x) = p(vt) = a^T B^d(t)$ on a m-dimensional simplex Δ with vertices v has integral*

$$\int_\Delta p(x)\,\mathrm{d}x = \frac{1}{m!\binom{d+m}{m}} \sum_\alpha a_\alpha. \tag{32}$$

2.2.1. The de Casteljau algorithm

The de Casteljau algorithm (Algorithm 1) evaluates polynomials in the Bernstein basis by the stable efficient process of iterated linear interpolation.

Algorithm 1. One-dimensional de Casteljau

INPUT: A one-dimensional polynomial

$$p(t) = a_0 B_0^d(t) + a_1 B_1^d(t) + \cdots + a_d B_d^d(t) =: a^T B^d(t) \tag{33}$$

in the Bernstein basis, and an evaluation point $t = (t_0, t_1 = 1 - t_0)$ in barycentric coordinates on $[0, 1]$.

for j in $0, \ldots, d$
  $a_{d-j,j}^0 := a_j$
end for
for j in $0, \ldots, d-1$
  for i in $0, \ldots, d-j$
  $a_{d-j-i,i}^{j+1} := t_0 a_{d-(j-1)-i,i}^j + t_1 a_{d-(j-1)-(i+1),i+1}^j$
  end for
end for

OUTPUT: $a_{0,0}^d = p(t)$.

Algorithm 1 proceeds from left to right through the columns of the following difference table:

$$\begin{array}{ccccccccc}
a_{d,0}^0 & \to & a_{d-1,0}^1 & \to & \cdots & \to & a_{1,0}^{d-1} & \to & a_{0,0}^d \\
 & \nearrow & & \nearrow & \cdots & \nearrow & & \nearrow & \\
a_{d-1,1}^0 & \to & a_{d-2,1}^1 & \to & \cdots & \to & a_{0,1}^{d-1} & & \\
 & \nearrow & & \nearrow & \cdots & \nearrow & & & \\
a_{d-2,2}^0 & \to & a_{d-3,2}^1 & \to & \cdots & & & & \\
\vdots & & \vdots & & \vdots & & & & \\
 & \nearrow & & \nearrow & & & & & \\
a_{1,d-1}^0 & \to & a_{0,d-1}^1 & & & & & & \\
 & \nearrow & & & & & & & \\
a_{0,d}^0 & & & & & & & &
\end{array}$$

It can be arranged to overwrite a single array in place (proceeding down each column), and generalizes (Algorithm 2) to evaluate n-dimensional Bernstein polynomials in $O(d^n)$ storage and $O(d^{2n})$ time.

2.2.2. Subdivision

The intermediate results a^j_α of the de Casteljau algorithm for $p(t)$ are also the Bernstein coefficients of p, in the Bernstein bases over the subsimplices generated by subdividing the original simplex at the common vertex t. Every a^j_α for which at least one component of α is zero becomes a coefficient on one or more of the subdivided simplices.

Algorithm 2. Multidimensional de Casteljau

INPUT: The coefficients of a degree-d polynomial

$$p(t) = \sum_{|\alpha|=d} a_\alpha B^d_\alpha(t) = a^T B^d(t) \tag{34}$$

in the m-dimensional Bernstein basis on a m-dimensional simplex Δ with vertices $v_0, \ldots, v_m \in R^n$, and the barycentric coordinates t of a point

$$x = vt = \sum_{j=0}^{m} t_j v_j \in \Delta. \tag{35}$$

for α **such that** $|\alpha| = d$
  $a^0_\alpha := a_\alpha$
end for
for j **in** $1, \ldots, d$
  for α **such that** $|\alpha| = d - j + 1$
  $a^j_\alpha = \sum_{k=0}^{m} t_k a^{j-1}_{\alpha+e_k}$
  end for
end for

OUTPUT: $a^d_0 = p(t)$

2.3. B-spline kernels

Approximate Gaussian smoothing kernels built from piecewise polynomials form a convenient basis for B-spline curves and surfaces [13], and can be defined by repeated autoconvolution of the characteristic function of an interval.

Definition 8. One-dimensional degree-d B-spline kernels ρ^d_h with width h are defined by

$$\rho^0_h(x) := \frac{1}{h}\chi_{[-\frac{h}{2},\frac{h}{2}]}(x) = \begin{cases} 0 & \text{if } |x| \geqslant \frac{h}{2}, \\ \frac{1}{h} & \text{if } |x| < \frac{h}{2}, \end{cases} \tag{36}$$

$$\rho^d_h(x) := \rho^{d-1}_h \star \rho^0_h(x), \tag{37}$$

where $\star$ denotes the usual convolution $f \star g(x) = \int_{-\infty}^{\infty} f(x-y)g(y)\mathrm{d}y$.

By the convolution theorem, the Fourier coefficients

$$\widehat{\rho^0_h}(k) = \int_{-\pi}^{\pi} \rho^0_h(x) e^{-\imath kx}\,\mathrm{d}x = \frac{1}{h}\int_{-h/2}^{h/2} e^{-\imath kx}\,\mathrm{d}x = \mathrm{sinc}(kh/2), \tag{38}$$

$$\widehat{\rho^d_h}(k) = \left(\widehat{\rho^0_h}\right)^{d+1} = (\mathrm{sinc}(kh/2))^{d+1} \tag{39}$$

are powers of sinc functions $\mathrm{sinc}(x) := \sin(x)/x$. Convolution of degree d_1 and d_2 piecewise polynomials produces a piecewise polynomial of degree $d_1 + d_2$. Since each integration also adds one order of global differentiability, ρ^d_h is a degree-d piecewise-polynomial function with $d-1$ continuous derivatives. Products of one-dimensional kernels produce multidimensional kernels with similar properties.

Definition 9. Multidimensional B-spline kernels of degree d and width h are given by

$$\rho_h^d(x) := \rho_h^d(x_1)\cdots\rho_h^d(x_n) \tag{40}$$

for $x = (x_1,\ldots,x_n) \in R^n$.

2.3.1. Recurrence relation and derivatives

The convolution of (37) may be evaluated by repeated linear interpolation, leading to the recurrence relation

$$\rho_h^d(x) = \frac{\rho_h^{d-1}\left(x+\frac{h}{2}\right)\left(x-\frac{d+1}{2}h\right)}{hd} + \frac{\rho_h^{d-1}\left(x-\frac{h}{2}\right)\left(\frac{d+1}{2}h-x\right)}{hd}. \tag{41}$$

Kernel derivatives thus satisfy the recurrence

$$\frac{\mathrm{d}\rho_h^d}{\mathrm{d}x}(x) = \frac{1}{h}\rho_h^{d-1}\left(x+\frac{h}{2}\right) - \frac{1}{h}\rho_h^{d-1}\left(x-\frac{h}{2}\right). \tag{42}$$

3. The GNUFFT

Let

$$f(\varphi) := \sum_{j=1}^{N}\int_{\Delta_j} p_j(t)\varphi(x)\mathrm{d}x \tag{43}$$

be an SPD with simplices $\Delta_j = \{x = v_j t | t \in B\}$ contained in the standard cube C. The Fourier coefficients of f are (Definition 5)

$$\widehat{f}(k) = \int_C f(x)e_k(x)\,\mathrm{d}x = \sum_{j=1}^{N}|\Delta_j|\int_B p_j(t)e_k(v_j t)\mathrm{d}t, \quad k\in\mathbb{Z}^n, \tag{44}$$

where v_j is the n by $m+1$ vertex matrix of Δ_j.

3.1. Overview

The GNUFFT approximates $\widehat{f}(k)$ for $k\in\mathbb{Z}^n$ with $|k| := \max(|k_1|,\ldots,|k_n|) \leqslant K$ by the Fourier coefficients of the smoothed function $\rho_h^d \star f$, then removes the smoothing in Fourier space. Convolution in real space is equivalent to multiplication in Fourier space, so

$$\widehat{f}(k) = \frac{1}{\widehat{\rho_h^d}(k)}\int_C e_k(x)\int_{R^n}\rho_h^d(x-y)f(y)\,\mathrm{d}y\,\mathrm{d}x \tag{45}$$

$$= \frac{1}{\widehat{\rho_h^d}(k)}\int_C e_k(x)\sum_{j=1}^{N}\int_{\Delta_j}\rho_h^d(x-t)p_j(t)\,\mathrm{d}t\,\mathrm{d}x. \tag{46}$$

The inner integral over each simplex Δ_j is exactly evaluated by a Q-point quadrature rule with points t_{jq} and weights w_{jq}, yielding

$$\widehat{f}(k) = \frac{1}{\widehat{\rho_h^d}(k)}\int_C e_k(x)\sum_{j=1}^{N}\sum_{q=1}^{Q} w_{jq}\rho_h^d(x-v_j t_{jq})p_j(t_{jq})\,\mathrm{d}x. \tag{47}$$

The outer integral over x is then approximated by $(2Ks)^n$-point oversampled trapezoidal quadrature with uniform points $c\in C$ and spacing $h = \pi/Ks$, yielding a finite sum with $(2Ks)^n NQ$ terms:

$$\widehat{f}(k) = \frac{1}{\widehat{\rho_h^d}(k)}h^n\sum_c e_k(c)\sum_{j=1}^{N}\sum_{q=1}^{Q} w_{jq}\rho_h^d(c-v_j t_{jq})p_j(t_{jq}) + \epsilon_T(k). \tag{48}$$

Once the localized sum over j is evaluated at each c, the sum over c can be evaluated efficiently by a standard FFT for wavenumbers with $|k| \leqslant Ks$. The process is summarized as Algorithm 3.

Algorithm 3. Geometric nonuniform fast Fourier transform

INPUT: A smoothing degree d, Fourier mode bound $K \geqslant 1$, real oversampling factor s, and an N-simplex SPD

$$f(\varphi) := \sum_{j=1}^{N} \int_{\Delta_j} p_j(t)\varphi(t)\mathrm{d}t$$

(**0**) Precomputation: Compute lookup tables and simplex quadrature formulas.
(**1**) Compute the smoothed function $f \star \rho_h^d$ on a regular grid:
  (**1.1**) Subdivide each simplex Δ carrying polynomial p into subsimplices $\Delta^\#$ carrying $p^\#$ subordinate to the oversampled grid of $(2Ks)^n$ cells.
  (**1.2**) Build a quadrature scheme for each subsimplex $\Delta^\#$ exact for polynomials of degree $\deg(p) + d$.
  (**1.3**) For each subsimplex $\Delta^\#$, evaluate the contribution to $f \star \rho_h^d$ at each uniform grid point due to integration over $\Delta^\#$.
(**2**) Compute the FFT $\widehat{\rho_h^d \star f}$ for $|k| \leqslant Ks$ by an n-variate FFT of size $(2Ks)^n$
(**3**) Divide $\widehat{\rho_h^d \star f} = \widehat{\rho_h^d}\widehat{f}$ by $\widehat{\rho_h^d}$, yielding $\widehat{f}$.

OUTPUT: $\widehat{f}(k)$ for $|k| \leqslant K$.

For zero-dimensional point sources, $Q = 1$ and the GNUFFT becomes a NUFFT based on the B-spline kernel [10]. Convolution trivially shifts and scales the kernel. For higher-dimensional simplices, nontrivial quadratures are required. Algorithm 4 below efficiently subdivides each simplex into subsimplices, each contained within a single grid cell where the kernel is a polynomial. Quadrature with degree of exactness $\deg(p_j) + d$ over each subsimplex of Δ_j then exactly integrates the product of kernel and density. Here the choice of a piecewise-polynomial B-spline kernel is convenient, rather than the usual Gaussian smoothing kernels employed in [14].

In the final step, the GNUFFT divides $\widehat{\rho_h^d \star f}$ by the transform $\widehat{\rho_h^d}$ of the smoothing kernel, which amplifies the errors in coefficients with high $|k|$. Oversampling the FFT by a factor $s > 1$ controls this amplification.

Subdivision takes $O(N(2Ks)^n)$ time. Quadrature takes $O((2Ks)^n(d + \deg(p))^m)$ time, where m is the dimension of a typical simplex in the SPD. The FFT itself takes $O(n(2Ks)^n \log(2Ks))$. Deconvolution takes $O((2Ks)^n)$ time. For large problem sizes the running time is dominated by quadrature, yielding an $O((2Ks)^n(d + \deg(p))^m)$ algorithm.

3.2. Subdivision quadrature

The GNUFFT integrates over each simplex Δ_j by (1) replacing the SPD f with an equivalent SPD $f^\#$, supported on simplices $\Delta^\#$ each contained in a single grid cell, (2) generating a quadrature scheme on each $\Delta^\#$ which exactly integrates polynomials of degree $\deg(p) + d$, and (3) summing the contributions $\rho_h^d \star (p^\# \Delta^\#)$.

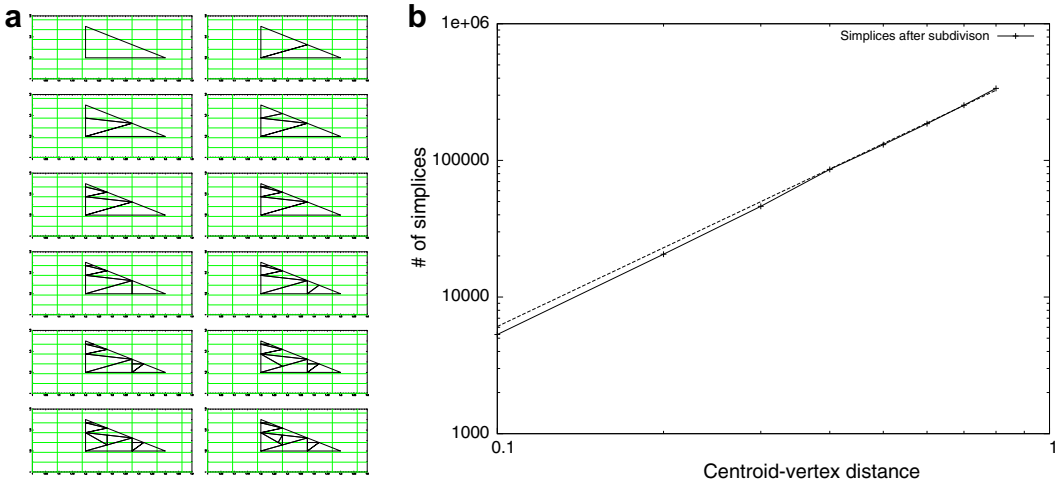


Fig. 2. (a) The first 11 steps of subdivision by Algorithm 4. (b) The number of subtriangles produced by applying Algorithm 4 to a single large triangle scales linearly with the triangle area.

3.2.1. Subdivision to a regular grid

The SPD is smoothed by convolution with a B-spline kernel ρ_h^d built of polynomials on cubes of width h. The smoothed function $\rho_h^d \star f$ is sampled at uniform grid points $c \in C$. If the simplices Δ of the SPD are subdivided into subsimplices $\Delta^\#$, each of which lies entirely within a single uniform grid cell, then quadrature of polynomial precision $d + \deg(p)$ on each $\Delta^\#$ will exactly evaluate the convolution $\rho_h^d \star (p^\# \Delta^\#)$. Algorithm 4 carries out this subdivision. Fig. 2(a) illustrates its operation on a small triangle that cuts a few grid cells.

When Algorithm 4 terminates, every simplex has been successfully subdivided. Each subdivision step creates simplices which have an L^2-norm edge length that is strictly smaller than that of the parent simplex. Thus eventual termination is guaranteed. Ideally, a subdivision algorithm should produce as few simplices as possible. However, it is likely that finding a minimal subdivision is NP-hard, implying that deterministic fast algorithms may not exist. The heuristic Algorithm 4 works extremely well in practice. Its efficiency was tested on equilateral plane triangles Δ of increasing size, with cell size $h = \frac{1}{512}$. Fig. 2(b) exhibits the total number of subsimplices $\Delta^\#$ produced by Algorithm 4 vs. the centroid-vertex distance of Δ, together with a Levenberg–Marquardt fit with exponent 1.9.

Algorithm 4. Subdivision to grid

INPUT: A list $\mathscr{L} = \{(\Delta, p)\}$ of simplices Δ carrying polynomials p.

(**1**) Create a new list $\mathscr{L}^\# = \{(\Delta, p, b)\}$, where b is a binary variable that is either "complete" or "incomplete" for each simplex.
(**2**) Mark all simplices in $\mathscr{L}^\#$ incomplete.
(**3**) **While $\mathscr{L}^\#$ contains at least one incomplete simplex,**
  (**3.1**) Let Δ be an incomplete simplex in $\mathscr{L}^\#$ carrying polynomial p
  (**3.2**) Let e be the longest edge of Δ that crosses a grid plane. If no such edge exists, mark Δ complete, and goto 3.
  (**3.3**) Let j be the index of the coordinate along which e crosses the largest number of grid planes.
  (**3.4**) Subdivide (Δ, p) at the point where e crosses the grid plane of coordinate j that cuts e most nearly into equal pieces. Remove (Δ, p) from $\mathscr{L}^\#$ and return both new simplices to $\mathscr{L}^\#$, marked incomplete
(**4**) **End while.**

OUTPUT: A list $\mathscr{L}^\#$ of simplices $\Delta^\#$ carrying polynomials $p^\#$.

3.2.2. Quadrature on subsimplices

We efficiently generate a quadrature of degree $D = \deg(p^\#) + d$, for each subsimplex $\Delta^\#$. Each integrand $\rho_h^d(x - v^\# t)p^\#(t)$ is a polynomial on $\Delta^\#$, so many exact quadrature options are available. For example, the integrand can be evaluated at equidistant points, transformed to the Bernstein basis, and integrated exactly by Corollary 1. However, this exact approach is neither optimally efficient nor numerically stable.

For one-dimensional quadrature, optimal Gaussian quadrature schemes can be computed efficiently. While such optimal schemes exist in higher dimensions, their computation is time-consuming. Instead, we iterate one-dimensional Gaussian schemes with the appropriate weight functions [15]. Iterated 1-D quadrature is exact, general and extremely stable [16], but suboptimally efficient. Performance can sometimes be improved by specialized quadrature schemes. For two-dimensional triangles, the symmetrical quadrature rules of [17] give speedups ranging from 12% for degree-5 smoothing up to 46% for degree-13 smoothing.

The resulting points $t_q^\#$ and weights $w_q^\#$ on $\Delta^\#$ depend only on the simplex dimension m and the polynomial degree $\deg(p_j) + d$, and are efficiently precomputed and stored. Thus subdivision quadrature yields a GNUFFT of the form

$$\widehat{f}(k) = \frac{1}{\widehat{\rho_h^d}(k)} h^n \sum_c e_k(c) \sum_{j=1}^{N} \sum_{\Delta^\# \subset \Delta_j} \sum_{q=1}^{Q} w_q^\# \rho_h^d(c - v_j^\# t_q^\#) p_j(t_q^\#) + \epsilon_T(k). \tag{49}$$

3.3. Speedup

In practice, integration over subsimplices consumes the bulk of the computing time. It can be factorized into geometric precomputation and runtime evaluation, to transform multiple distributions with the same geometry, by merging quadratures over subdivided simplices into quadratures over the original simplices. The polynomial densities

$$p_j(t) = a_j^T B^{\deg p_j}(t) \tag{50}$$

are independent of the subdivision, so (49) factorizes as

$$\widehat{f}(k) = \frac{1}{\widehat{\rho_h^d}(k)} h^n \sum_c e_k(c) \sum_{j=1}^{N} a_j^T \sum_{\Delta^\# \subset \Delta_j} \sum_{q=1}^{Q} w_q^\# \rho_h^d(c - v_j^\# t_q^\#) B^{\deg p_j}(t_q^\#) + \epsilon_T(k). \tag{51}$$

The inner sums over $\Delta^\#$ and q depend only upon the geometry of the SPD and the degrees of the p_j. The sum over c has $(2Ks)^n$ terms. They may thus be precomputed and stored, yielding a quadrature formula on Δ_j suitable for efficient smoothing of multiple distributions on a common geometry.

4. Error analysis

The GNUFFT approximates the Fourier coefficients by the trapezoidal rule, incurring an aliasing error. This error is amplified in the deconvolution step, when the GNUFFT divides the transformed function by the small wings of $\widehat{\rho_h^d}$. Each error is bounded and controlled by the oversampling factor s and the smoothing kernel degree d.

4.1. Aliasing error

The exact Fourier transform $\widehat{f_s}$ of the smoothed SPD $f_s = f \star \rho_h^d$ is approximated by the oversampled trapezoidal sums

$$\widetilde{f_s}(k) = h^n \sum_c f_s(c) e^{-\imath k^T c} \tag{52}$$

$$= (2Ks)^{-n} \sum_c \sum_{m \in Z^n} \widehat{f_s}(m) e^{\imath m^T c - \imath k^T c} \tag{53}$$

$$= \sum_m \widehat{f_s}(k + 2Ksm) \tag{54}$$

$$= \widehat{f_s}(k) + \epsilon_a(k), \tag{55}$$

where we have used the Fourier series inversion formula

$$f_s(x) = (2\pi)^{-n} \sum_k \widehat{f_s}(k) e^{\imath k^T x}$$

and the discrete orthogonality relation

$$(2Ks)^{-n} \sum_c e^{\imath k^T c} = \begin{cases} 1 & \text{if } k \equiv 0 \pmod{2Ks} \\ 0 & \text{otherwise} \end{cases}$$

of the Fourier basis functions. The aliasing error $\epsilon_a(k)$ is bounded by

$$|\epsilon_a(k)| = \left| \sum_{0 \neq m \in Z^n} \widehat{\rho_h^d}(k + 2Ksm) \widehat{f}(k + 2Ksm) \right| \tag{56}$$

$$\leqslant F \sum_{0 \neq m \in Z^n} \left| \widehat{\rho_h^d}(k + 2Ksm) \right| \tag{57}$$

$$\leqslant F \sum_{0 \neq m \in Z^n} \prod_{j=1}^{n} \left| \operatorname{sinc}\left((k_j + 2Ksm_j)h/2\right) \right|^{d+1} \tag{58}$$

$$= F \sum_{0 \neq m \in Z^n} \prod_{j=1}^{n} \left| \operatorname{sinc}\left(\pi m_j + \frac{\pi k_j}{2Ks}\right) \right|^{d+1}, \tag{59}$$

where $m = (m_1, \ldots, m_n) \in Z^n$ and

$$|\widehat{f}(k)| \leqslant F := \sum_{j=1}^{N} \int_{\Delta_j} |p_j(x)| \, \mathrm{d}x. \tag{60}$$

Since

$$|\operatorname{sinc}(\pi\mu + \theta)| \leqslant \frac{\sin \pi/2s}{\pi |\,|\mu| - 1/2s|} \quad \text{for } \mu \in Z \quad \text{and} \quad |\theta| < \pi/2s < \pi/2$$

and the output wavenumber k has components $|k_j| \leqslant K$,

$$|\epsilon_a(k)| \leqslant F \sum_{m \neq 0} \prod_{j=1}^{n} \left(\frac{\sin \pi/2s}{\pi ||m_j| - 1/2s|} \right)^{d+1} = F(\operatorname{sinc}(\pi/2s))^{n(d+1)} g_{d+1}(s),$$

where

Table 2
Error amplification factor σ for deconvolution.

s	$n=1$		$n=2$		$n=3$	
	d	σ	d	σ	d	σ
2	3	6.09×10^{0}	3	3.71×10^{1}	3	2.26×10^{2}
	5	1.50×10^{1}	5	2.26×10^{2}	5	3.39×10^{3}
	7	3.71×10^{1}	7	1.37×10^{3}	7	5.09×10^{4}
	9	9.15×10^{1}	9	8.36×10^{3}	9	7.65×10^{5}
	11	2.26×10^{2}	11	5.09×10^{4}	11	1.15×10^{7}
3	3	2.14×10^{0}	3	4.57×10^{0}	3	9.77×10^{0}
	5	3.13×10^{0}	5	9.77×10^{0}	5	3.05×10^{1}
	7	4.57×10^{0}	7	2.09×10^{1}	7	9.55×10^{1}
	9	6.68×10^{0}	9	4.47×10^{1}	9	2.98×10^{2}
	11	9.77×10^{0}	11	9.55×10^{1}	11	9.33×10^{2}

$$g_d(s) := \sum_{m\neq 0}\prod_{j=1}^{n}|2s|m_j|-1|^{-d} = (1+S)^n - 1.$$

The one-dimensional reciprocal power sum

$$S := 2\sum_{m=1}^{\infty}\frac{1}{(2ms-1)^d} \leqslant \frac{2}{(2s-1)^d}\sum_{m=1}^{\infty} m^{-d} \leqslant \frac{2.1}{(2s-1)^d} \leqslant 1$$

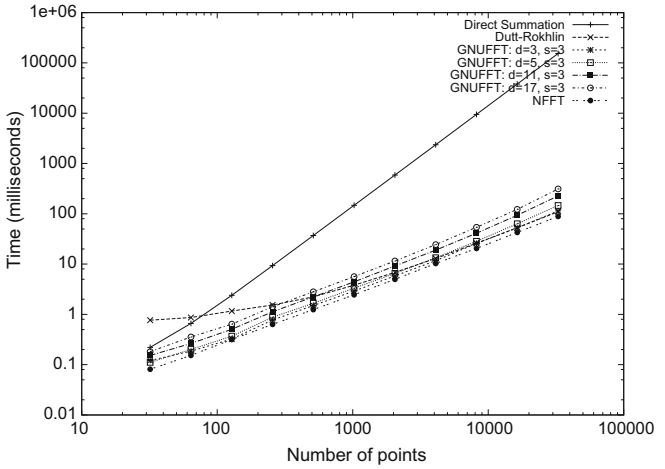
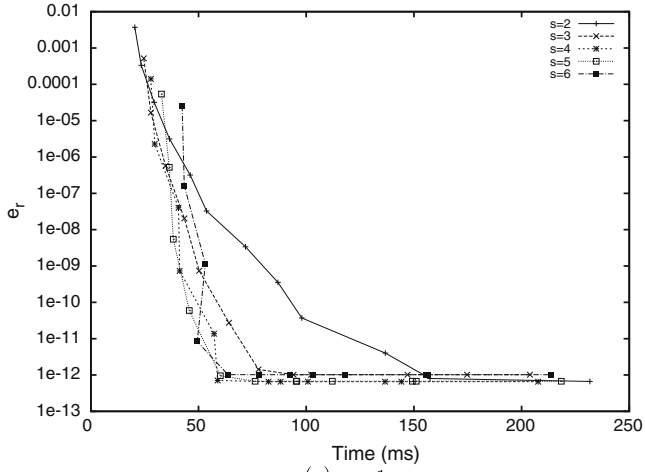


Fig. 3. Times for GNUFFT, direct summation, NFFT, and the Dutt–Rokhlin algorithm.

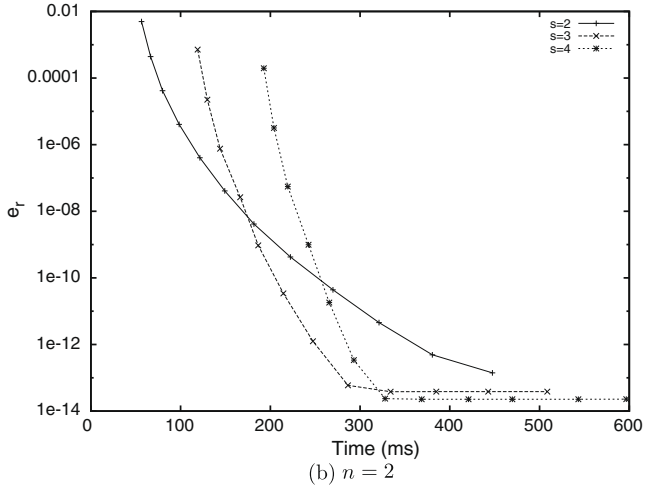
Table 3
CPU time for the GNUFFT of 16384 points in one dimension, with various kernel degrees d and oversampling factors s.

s	FFT size	d	τ_c	τ_f	τ_t	e_r	Error bound
2	65536	3	20.6	10.8	57.7	3.65×10^{-3}	1.09×10^{1}
3	98304	3	21.5	18.2	75.1	5.11×10^{-4}	1.41×10^{0}
4	131072	3	22.4	22.5	89.9	1.40×10^{-4}	3.68×10^{-1}
2	65536	7	35.8	9.0	76.5	3.19×10^{-5}	1.35×10^{-1}
3	98304	7	38.4	16.9	96.5	5.69×10^{-7}	2.26×10^{-3}
4	131072	7	38.0	20.9	109.4	3.97×10^{-8}	1.53×10^{-4}
2	65536	11	51.7	9.0	99.0	3.23×10^{-7}	1.66×10^{-3}
3	98304	11	53.6	16.5	116.8	7.48×10^{-10}	3.62×10^{-6}
4	131072	11	52.1	20.9	128.9	2.55×10^{-11}	6.38×10^{-8}
2	65536	15	64.4	9.0	116.3	3.48×10^{-9}	2.05×10^{-5}
3	98304	15	68.8	16.6	138.0	6.80×10^{-11}	5.79×10^{-9}
4	131072	15	66.7	21.0	149.7	2.15×10^{-11}	2.66×10^{-11}

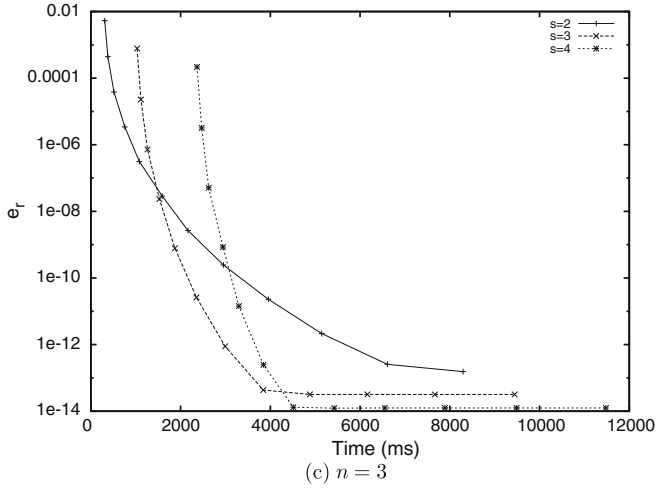
for oversampling factors $s \geqslant 1.1$, smoothing kernel degrees $d \geqslant 4$, and dimensions $1 \leqslant n \leqslant 3$. Thus by the finite geometric series formula, the aliasing error bound



(a) $n = 1$



(b) $n = 2$



(c) $n = 3$

Fig. 4. $N = 4096$ points with wavenumbers $|k| \leqslant K := 4096$.

$$|\epsilon_a(k)| \leqslant F\,\frac{2^{n+1}\mathrm{sinc}(\pi/2s)^{n(d+1)}}{(2s-1)^{d+1}}$$

can be made as small as desired by increasing the oversampling factor s and the degree d of the smoothing kernel.

4.2. Error amplification

Deconvolution divides each mode $\widehat{f}_s(k) = \widehat{\rho_h^d \star f}(k)$ by $\widehat{\rho_h^d}(k)$. Since $\mathrm{sinc}(x)$ is a decreasing function on the interval $0 < x < \pi/2$,

$$|\widehat{\rho_h^d}(k)| = \prod_{j=1}^{n} |\mathrm{sinc}(k_j h/2)^{d+1}| \geqslant |\mathrm{sinc}(\pi/2s)|^{n(d+1)} \quad \text{for } |k_j| \leqslant K. \tag{61}$$

Thus deconvolution amplifies aliasing errors by a factor no larger than $\sigma = \mathrm{sinc}(\pi/2s)^{-n(d+1)}$. Since this factor already appears in the aliasing error bound, the error $\epsilon_T(k)$ in the final result $\widehat{f}(k)$ is controlled by

$$|\epsilon_T(k)| = \left|\frac{\epsilon_a(k)}{\mathrm{sinc}(\pi/2s)^{n(d+1)}}\right| \leqslant F\,\frac{2^{n+1}}{(2s-1)^{d+1}},$$

which can be made arbitrarily small by increasing the oversampling factor and the smoothing degree. Other errors such as roundoff are amplified by at worst the factors σ shown in Table 2.

5. Numerical results

A version of the GNUFFT algorithm has been implemented in the C programming language, and its efficiency and accuracy have been extensively studied in a gallery of complex test cases. Test cases have been designed with approximately equal input and output sizes to measure the algorithm speed realistically.

The tables below report various timing measures:

τ_c = CPU time required for convolution.
τ_f = CPU time for a standard oversampled FFT of size $(2Ks)^n$.
τ_t = Total CPU time.

All timings correspond to wall-clock time on a 2.16 GHz Intel Core 2 Duo iBook with 2.5 GB RAM, under Mac OS X 10.4.11, with the open source compiler `gcc 4.0.1` and optimization flag `-O3`.

We report Fourier coefficient errors

$$e_a = \|\widetilde{f} - \widehat{f}\|_2 = \left(\sum_{|k|\leqslant K} |\widetilde{f}(k) - \widehat{f}(k)|^2\right)^{1/2}, \quad e_r = \frac{e_a}{\|\widehat{f}\|_2} \tag{62}$$

are approximate absolute and relative errors that are not sensitive to near-zero components of $\widehat{f}$. For point and line segment distributions in $n = 1$ ambient dimension, $\widehat{f}$ is evaluated directly to measure errors. In more complicated cases with $m \geqslant 1$ or

Table 4
CPU performance for the GNUFFT applied to a sawtooth function in one dimension.

s	FFT size	d	τ_c	τ_f	τ_t	e_r	Error bound
1	2048	3	5.1	0.3	44.8	5.71×10^{-3}	3.14×10^{2}
2	4096	3	7.9	0.7	40.7	2.64×10^{-5}	3.87×10^{0}
3	6144	3	12.9	1.3	63.0	1.88×10^{-6}	5.02×10^{-1}
4	8192	3	15.9	1.3	81.7	3.12×10^{-7}	1.31×10^{-1}
1	2048	7	10.5	0.3	27.0	4.15×10^{-3}	3.14×10^{2}
2	4096	7	21.2	0.5	54.2	2.55×10^{-7}	4.78×10^{-2}
3	6144	7	34.43	0.9	84.8	2.60×10^{-9}	8.03×10^{-4}
4	8192	7	41.9	1.1	109.2	1.21×10^{-10}	5.44×10^{-5}
1	2048	11	19.9	0.3	37.6	3.40×10^{-3}	3.14×10^{2}
2	4096	11	39.6	0.5	74.1	2.63×10^{-9}	5.90×10^{-4}
3	6144	11	65.8	0.9	118.0	4.29×10^{-12}	1.28×10^{-6}
4	8192	11	80.1	1.1	149.5	3.20×10^{-13}	2.27×10^{-8}
1	2048	15	32.3	0.3	53.1	2.95×10^{-3}	3.14×10^{2}
2	4096	15	64.7	0.5	102.3	3.38×10^{-11}	7.29×10^{-6}
3	6144	15	107.9	0.9	163.2	1.47×10^{-12}	2.06×10^{-9}
4	8192	15	130.9	1.1	203.7	4.65×10^{-13}	9.44×10^{-12}

$n \geqslant 2$, results are compared against another GNUFFT with smoothing degree $d = 15$ and oversampling factor $s = 4.4$, sufficient for full machine accuracy.

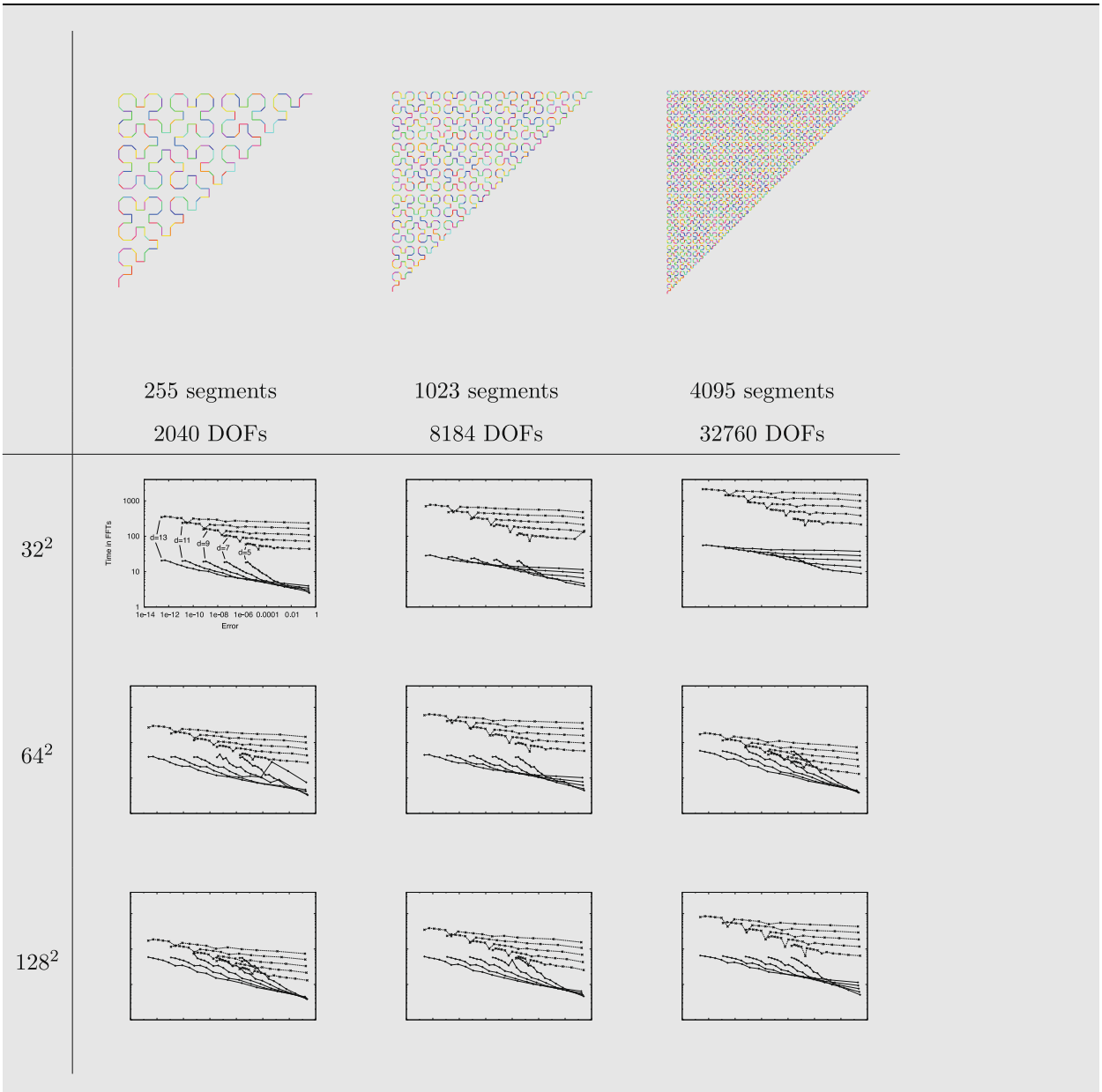
Times are reported as multiples of one FFT that returns the same number of Fourier coefficients as the GNUFFT in question. On our test iMac, a 32^2 FFT takes 1.1 ms, a 64^2 FFT 2.1 ms, and a 128^2 FFT 5.4 ms. In the $n = 3$ case on the same machine, a 16^3 transform takes 0.55 ms, while a 32^3 transform takes 3.9 ms.

5.1. Simplex dimension $m = 0$

First consider SPDs

$$f(x) = \sum_{i=1}^{N} p_j \delta(x - x_j) \tag{63}$$

Table 5
Performance of the GNUFFT on 1D data in 2D.



supported on zero-dimensional points x_j which are uniformly distributed on $[0,2]^n$. The constant polynomials p_j are uniformly distributed on $[-1,1]$.

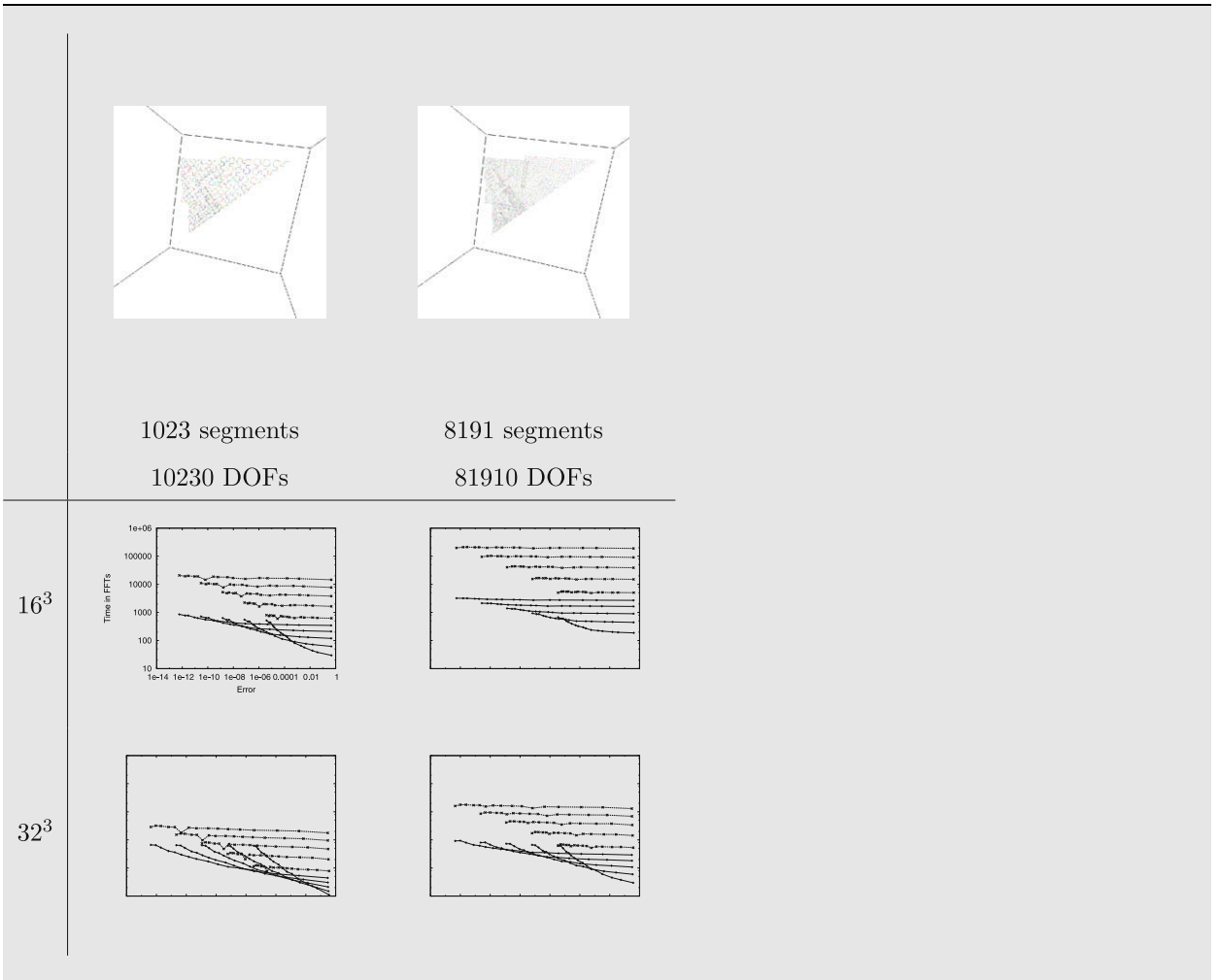
5.1.1. Space dimension $n = 1$

Input and output sizes are approximately balanced by computing all modes of wavenumber $|k| \leqslant K := N/2$ for N input points on $[0,2]$. Fig. 3 compares the performance of the GNUFFT to our implementations of direct summation and the Dutt–Rokhlin algorithm with error tolerance $\epsilon = 10^{-11}$, and to the publicly available NFFT library [10]. The four cases of GNUFFT plotted used oversampling $s = 3$ and smoothing degree $d = 3 (e_r \approx 5 \times 10^{-4}), d = 5 (e_r \approx 2 \times 10^{-5}), d = 11 (e_r \approx 7 \times 10^{-10})$, and $d = 17 (e_r \approx 4 \times 10^{-12})$. The GNUFFT runs at essentially the same speed as Dutt–Rokhlin despite its increased generality. NFFT is slightly faster; we attribute this to the library using a one-dimensional implementation for the $n = 1$ transform. The detailed results of Table 3 confirm the efficiency and accuracy of the algorithm in this setting. Fig. 4 shows the tradeoffs between time and accuracy over a wide range of oversampling factors s and for kernel degrees d ranging over $3, 5, 7, \ldots, 25$, for a test case with 4096 input points and output modes.

5.1.2. Space dimensions $n = 2$ and $n = 3$

To equalize input and output sizes, we compute wavenumbers $|k| \leqslant K := N^{1/n}$ with N points in $[0,2]^n$ for test cases in $n > 1$ dimensions. Tables 10 and 11 list results for 16,384 and 32,768 points with a variety of parameters. Fig. 4(b) and (c) shows tradeoffs between accuracy and time for 4096 points. Oversampling is more costly in higher dimensions, so oversampling factors higher than 4 were dropped.

Table 6
Performance of the GNUFFT on 1D data in three dimensions.



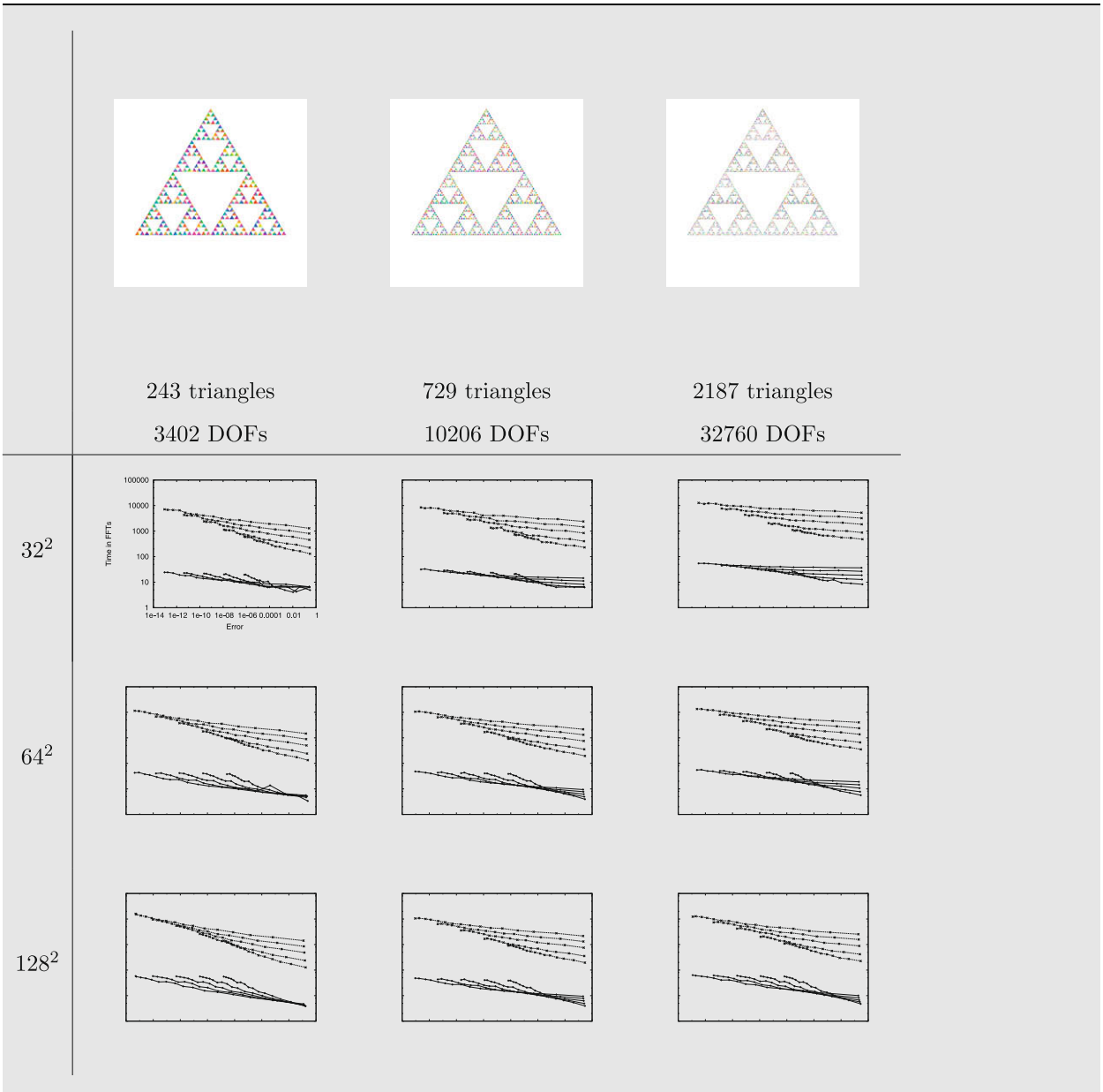
5.2. Simplex dimension $m = 1$

Next we consider the Fourier transform of SPDs supported on line segments, reminiscent of boundary integrals in two dimensions and curve singularities in three dimensions. The $N \times N$ discrete Fourier transform has $2N^2$ real degrees of freedom for input and output. A line segment supporting a complex linear distribution in n dimensions has $4 + 2n$ degrees of freedom. We balance input and output size by requiring the output to have the same number of degrees of freedom as the input.

5.2.1. Space dimension $n = 1$

A SPD in R supported by one-dimensional simplices is a piecewise polynomial function whose transform can be computed exactly. For example, the periodic sawtooth

Table 7
Performance of the GNUFFT on 2D data in two dimensions.



$$f(x) = -1 + x \tag{64}$$

has the exact transform

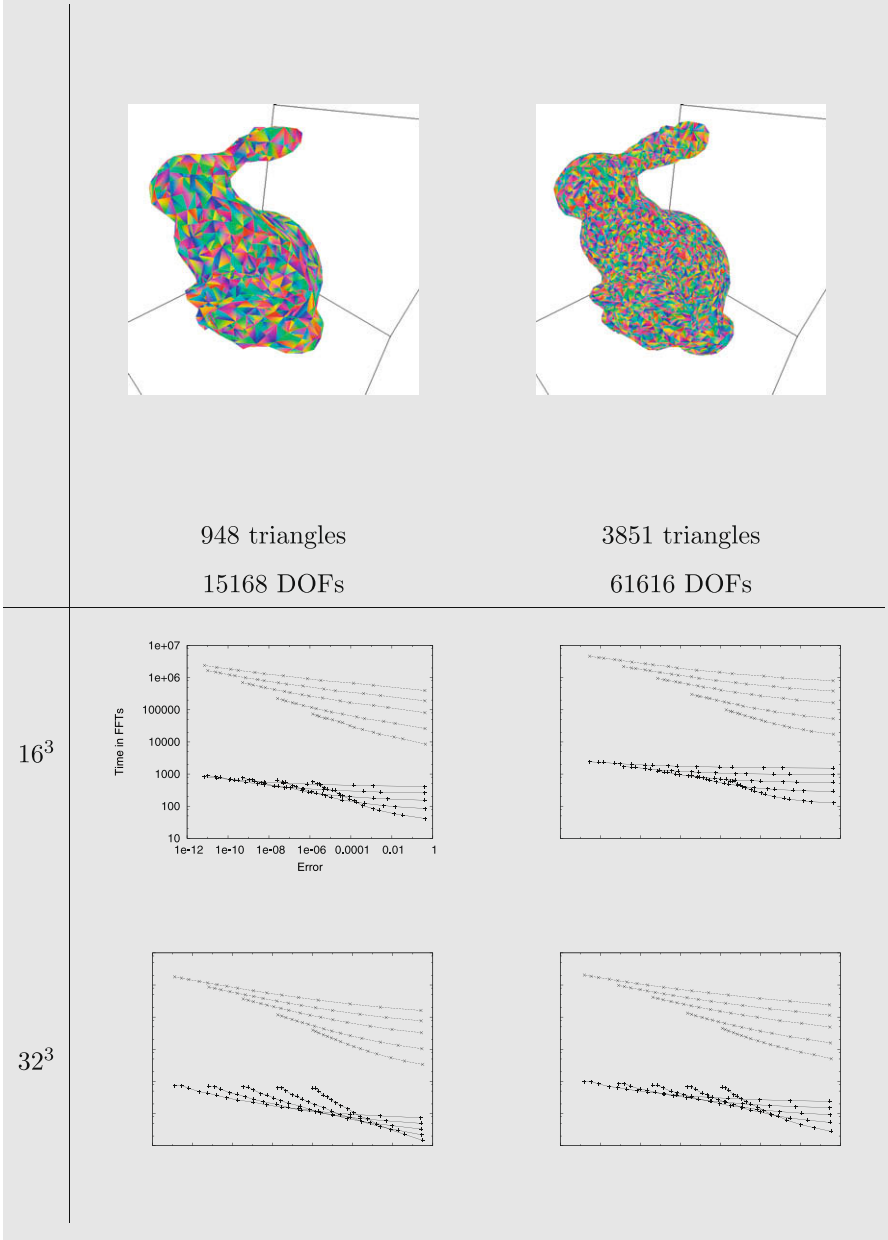
$$\widehat{f}(k) = \int_0^2 f(x) e^{-\pi \imath x k} dx = \frac{2\imath}{\pi k}. \tag{65}$$

GNUFFT errors for this input function are reported in Table 4.

5.2.2. Line segments in higher dimensions

A convenient set of test cases is provided by the sequence of piecewise linear curves whose limit is the Sierpinski curve [18], with a random linear complex polynomial on each segment (see Table 5).

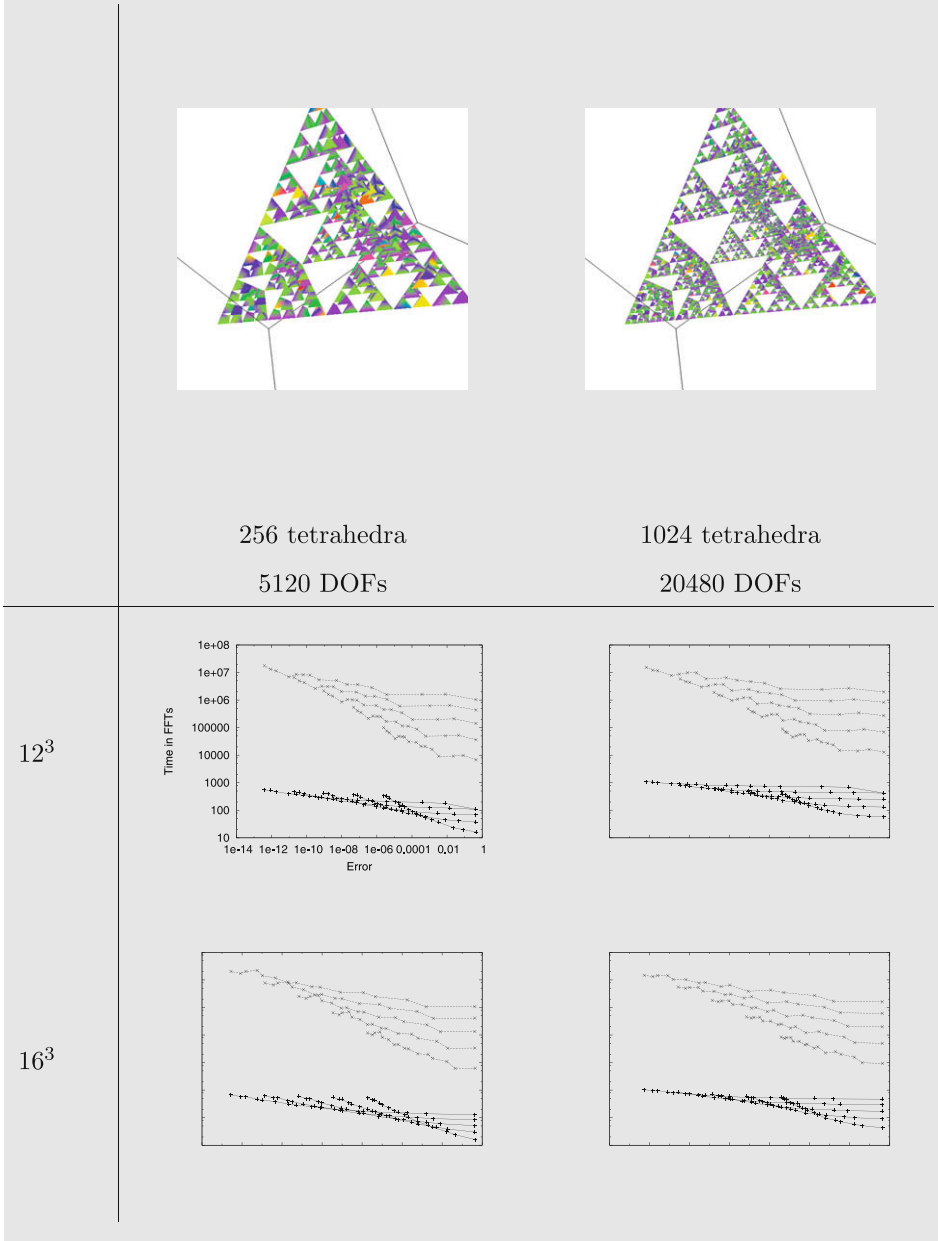
Table 8
Performance of the GNUFFT on 2D data in three dimensions.



Since GNUFFTs of output size $(2K)^2 = 32^2, 64^2$, and 128^2 have $2^{11}, 2^{13}$, and 2^{15} degrees of freedom respectively and a line segment in two dimensions supporting a complex linear polynomial has eight real degrees of freedom, input and output sizes are balanced with curves containing $2^8 - 1, 2^{10} - 1$, and $2^{12} - 1$ segments. The results are plotted in Table 5. The upper series of lines on each graph show precomputation time vs. error; the lower series of lines show computation time vs. error. Each line corresponds to a different smoothing kernel degree d ranging over $5, 7, \ldots, 13$, while the oversampling parameter s increases from 1.1 to 4.4 along each line. In these cases precomputation typically takes 100-1000 times as long as an equivalent uniform FFT, while computation typically takes 10–100 FFTs. After precomputation, each extra digit of accuracy costs about a 15% increase in computation time.

In three dimensions, 16^3 and 32^3 GNUFFTs yield 2^{13} and 2^{16} degrees of freedom respectively. Thus convenient test cases are provided by the same curves, displaced into three dimensions so that the curve climbs at a constant rate. Results are shown in Table 6. In three dimensions oversampling is much more expensive; the cost after precomputation is typically

Table 9
Performance of the GNUFFT on 3D data in three dimensions.



hundreds of FFTs. Additional accuracy is similarly more expensive; each additional digit in the balanced 32^3 case, for example, costs a 27% increase in time.

5.3. Distributions on triangles

A triangle supporting a complex linear polynomial in n dimensions has $6 + 3n$ degrees of freedom. For a grid size h a triangle subdivides to $O(h^{-2})$ subtriangles, making precomputation considerably more expensive than it is for line segments. Thus $n \times n$ Fourier coefficients balance the complexity of $n^2/6$ triangles supporting linear polynomials.

5.3.1. Triangles in two dimensions

A two-dimensional test case with two-dimensional triangles, is provided by a finite Sierpinski Gasket [19]. Results with $3^5, 3^6$, and 3^7 triangles are shown in Table 7. Each transform takes 10–100 FFTs after precomputation. Precomputation cost up to 10,000 FFTs in these trials, because quadrature schemes on triangles are expensive compared to line segments. In the 3^6 triangles case with a 64^2 transform, each additional digit of accuracy requires about a 21% increase in time.

5.3.2. Triangles in three dimensions

The Stanford bunny [20] is a readily-available triangulated surface with 948 or 3851 triangles. Results for 16^3 and 32^3 Fourier transforms are plotted in Table 8. Each transform requires time equivalent to a few hundred of these small FFTs, after precomputation costing tens of thousands to millions of FFTs. Each additional digit of accuracy costs about 21% increase in time for the 3851-triangle 32^3 transform.

5.4. Distributions on tetrahedra

The three-dimensional analog [19] of the Sierpinski gasket begins with an equilateral tetrahedron and recursively replaces each parent tetrahedron with four tetrahedra, each of which is the hull of one vertex of the parent and the midpoints

Table 10
CPU time for the GNUFFT of 16384 points in two dimensions.

s	FFT size	d	τ_c	τ_f	τ_t	e_r	Error bound
2	512^2	3	49.3	38.6	245.5	5.13×10^{-3}	4.42×10^{1}
3	768^2	3	58.9	126.6	529.9	7.20×10^{-4}	5.73×10^{0}
4	1024^2	3	67.2	170.8	843.6	1.95×10^{-4}	1.49×10^{0}
2	512^2	7	132.1	27.6	322.1	4.39×10^{-5}	5.46×10^{-1}
3	768^2	7	147.4	95.0	593.9	7.86×10^{-7}	9.17×10^{-3}
4	1024^2	7	158.1	148.7	926.2	5.42×10^{-8}	6.21×10^{-4}
2	512^2	11	269.3	28.3	468.4	4.35×10^{-7}	6.74×10^{-3}
3	768^2	11	283.6	95.3	752.6	1.02×10^{-9}	1.47×10^{-5}
4	1024^2	11	305.2	150.0	1090.4	3.83×10^{-11}	2.59×10^{-7}
2	512^2	15	449.1	27.4	651.0	4.60×10^{-9}	8.32×10^{-5}
3	768^2	15	466.4	95.7	929.9	1.06×10^{-10}	2.35×10^{-8}
4	1024^2	15	497.6	149.5	1283.4	3.37×10^{-11}	1.08×10^{-10}

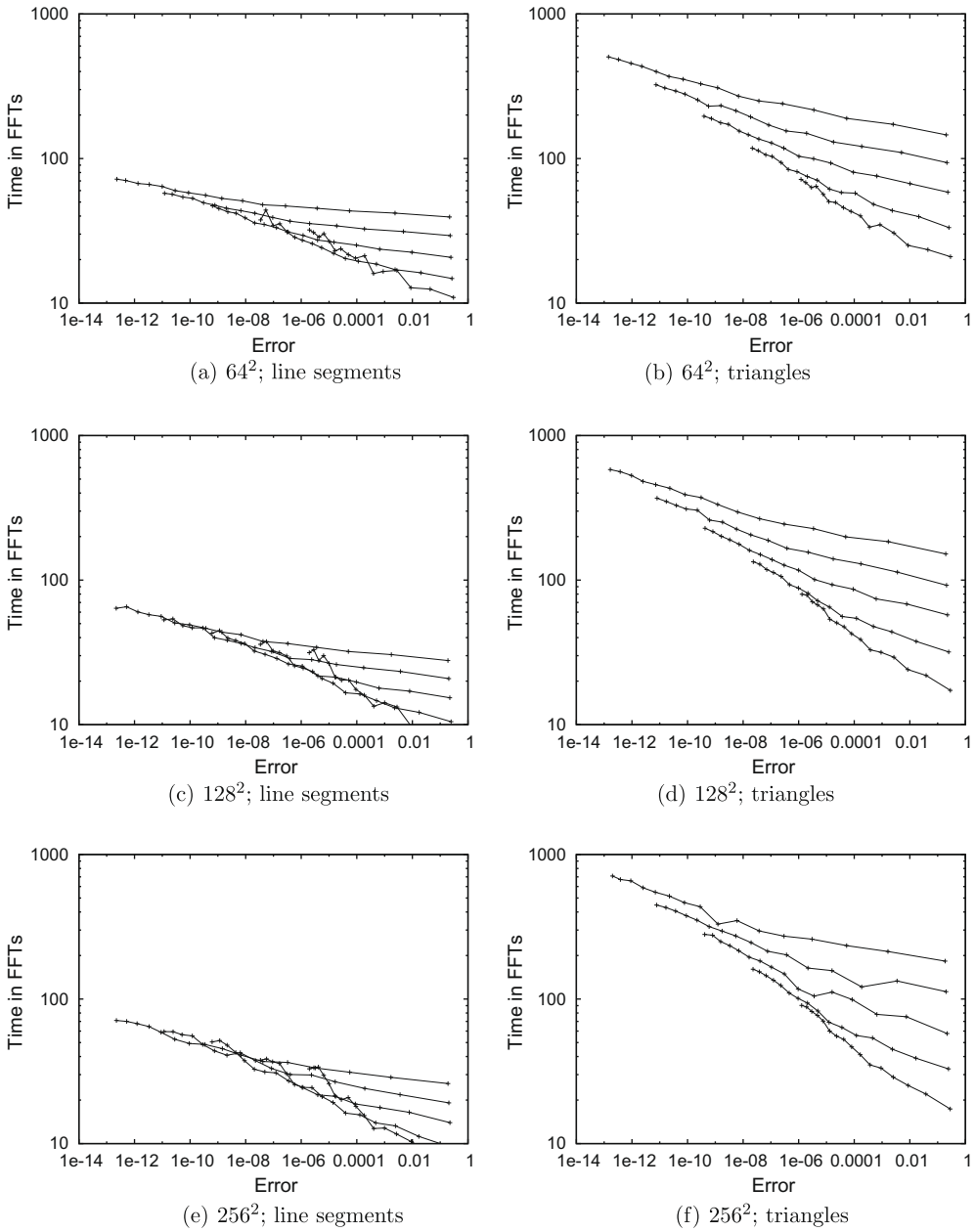
Table 11
CPU performance of the 3D GNUFFT on 32768 data points in three dimensions, returning 32^3 modes.

s	FFT size	d	τ_c	τ_f	τ_t	e_r	Error bound
2	128^3	3	363.5	492.8	2678.7	5.83×10^{-3}	2.47×10^{2}
3	192^3	3	468.3	2116.1	8712.3	8.27×10^{-4}	3.20×10^{1}
4	256^3	3	643.2	4082.4	19095.2	2.26×10^{-4}	8.33×10^{0}
2	128^3	7	2202.1	495.1	4558.4	4.64×10^{-5}	3.05×10^{0}
3	192^3	7	2309.1	2076.5	10606.6	8.47×10^{-7}	5.12×10^{-2}
4	256^3	7	2585.5	4073.7	21243	5.88×10^{-8}	3.47×10^{-3}
2	128^3	11	7264.3	495.2	9670.36	4.27×10^{-7}	3.76×10^{-2}
3	192^3	11	7212.3	2067.1	15649.3	1.03×10^{-9}	8.19×10^{-5}
4	256^3	11	7607.5	4112.4	26644.5	4.73×10^{-11}	1.44×10^{-6}
2	128^3	15	16781.2	497.7	19187.9	4.20×10^{-9}	4.64×10^{-4}
3	192^3	15	16536.7	2060.9	24936.3	1.38×10^{-10}	1.31×10^{-7}
4	256^3	15	17208.8	4133.7	36179.9	4.37×10^{-11}	6.02×10^{-10}

of the edges connected to that vertex. Results are plotted in Table 9 and . Transforms typically take hundreds of FFTs, while precomputation is about an order of magnitude more expensive than in the 3D triangle case, requiring tens of millions of FFTs. Each additional digit of accuracy requires about 21% more time.

5.5. Small simplices without precomputation

The above results on segments, triangles, and tetrahedra all precompute the smoothing operator. Without precomputation, the performance of the algorithm can be inconsistent, because the time required to smooth a given simplex depends on its size. A consistent test involves distributions of randomly placed simplices with edge lengths constrained to fall between 0.5 and 2.0 times the grid length. The resulting times are plotted in Figs. 5 and 6(a) through Fig. 5(f). In these small cases, computation times are tens of FFTs in two dimensions, hundreds of FFTs in three.



(a) 64^2; line segments (b) 64^2; triangles

(c) 128^2; line segments (d) 128^2; triangles

(e) 256^2; line segments (f) 256^2; triangles

Fig. 5. Extremely small simplices, no precomputation.

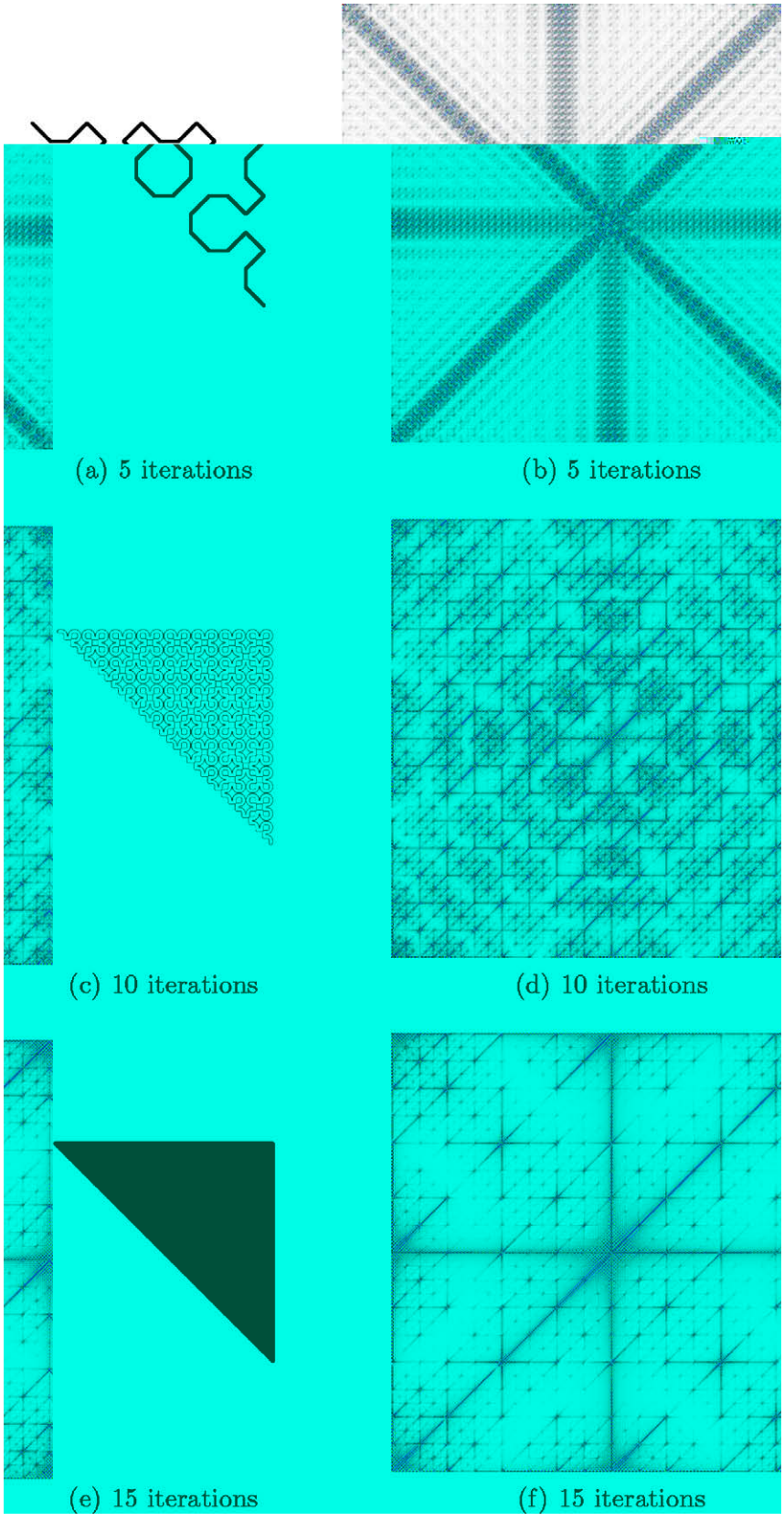


Fig. 6. Constant densities on approximate Sierpinski paths (left) and their transforms (right).

6. Future work

Several potential optimizations and extensions appear straightforward. The long precomputation times are partly due to the $O(\log N)$ insertion time into a B-tree data structure. A faster structure such as a hash map could store the smoothing operator in slightly more space and retrieve it faster, yielding shorter precomputation times. Smoothing with the B-spline kernel is left-invertible, so an inverse GNUFFT is possible. The present algorithm could easily be combined with Ying's sparse FFT [21] to evaluate a specified subset of Fourier modes, rather than the full box $[-K, K]^n$. Such computations are common in scattering theory.

Acknowledgments

We thank C. Epstein, W. Kahan, D. Bass, and T. Chen for valuable discussions. This work was partially supported by NSF Grant DMS-0512963, by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under Grant Number FA9550-05-1-0120, and by the University of California. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

References

[1] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19 (1965) 297–301.
[2] H.J. Levinson, Principles of Lithography, SPIE, 2001.
[3] P. Ewald, Die Berechnung optischer und elektrostatistischer Gitterpotentiale, Ann. Phys. 64 (1921) 253.
[4] J. Strain, Fast potential theory II: Layer potentials and discrete sums, J. Comput. Phys. 99 (1992) 251–270.
[5] D. Saintillan, E. Darve, E.S.G. Shaqfeh, A smooth particle-mesh Ewald algorithm for Stokes suspension flows: the sedimentation of fibers, Phys. Fluids 17 (2004) 1–21.
[6] J. Strain, Locally-corrected spectral methods and overdetermined elliptic systems, J. Comput. Phys. 224 (2007) 1243–1254.
[7] J. Tausch, A spectral method for the fast solution of boundary integral formulations of elliptic problems, in: C. Constanda, Z. Nashed, D. Rollins (Eds.), Integral Methods in Science and Engineering, Birkhauser, 2006, pp. 287–297.
[8] G. Beylkin, On the fast Fourier transform of functions with singularities, Appl. Comput. Harm. Anal. 2 (1995) 363–381.
[9] A. Dutt, V. Rokhlin, Fast Fourier transforms for nonequispaced data, SIAM J. Sci. Comput. 14 (6) (1993) 1368–1393.
[10] D. Potts, G. Steidl, Fast summation at nonequispaced knots by NFFTs, SIAM J. Sci. Comput. 24 (2003) 2013–2037.
[11] G.G. Lorentz, Bernstein Polynomials, Mathematical Expositions, vol. 8, University of Toronto Press, Toronto, 1953.
[12] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, Dover, 1965.
[13] C. de Boor, A Practical Guide to Splines, Applied Mathematical Sciences, vol. 27, Springer-Verlag, 2001. Revised ed..
[14] A. Dutt, V. Rokhlin, Fast Fourier transforms for nonequispaced data II, Appl. Comput. Harmon. Anal. 2 (1) (1995) 85–100.
[15] A.H. Stroud, D. Secrest, Gaussian Quadrature Formulas, Prentice-Hall, 1966.
[16] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, Textbook in Applied Mathematics, third ed., vol. 12, Springer-Verlag, New York, 2002.
[17] D.A. Dunavant, High degree efficient symmetrical gaussian quadrature rules for the triangle, Int. J. Numer. Methods Eng. 21 (6) (1985) 1129–1148.
[18] H. Sagan, Space-Filling Curves, Springer-Verlag, 1994.
[19] W. Sierpiński, Sur une courbe dont tout point est un point de ramification, C.R. Acad. Sci. Paris 160 (1915) 302–305.
[20] G. Turk, M. Levoy, Zippered polygon meshes from range images, in: Computer Graphics, SIGGRAPH, 1994, pp. 311–318.
[21] L. Ying, Sparse fourier transform via butterfly algorithm, SIAM J. Sci. Comput., in press.