Fast Fourier transforms of piecewise polynomials

John Strain

Department of Mathematics, University of California, 970 Evans Hall, Berkeley, California 94720-3840

Abstract

We introduce an efficient algorithm for evaluating the Fourier transform of piecewise-polynomial data on d-dimensional simplices in D-dimensional Euclidean space $\mathbf{R}^D$. It generalizes butterfly algorithms for pointwise ($d = 0$) nonuniform fast Fourier transforms, with new dimensional recurrences for exponential-polynomial moments. Error analysis and numerical comparisons with direct evaluation validate the efficiency and accuracy of the algorithm.

Keywords: Fourier transform, butterfly algorithm, spectral methods
PACS: 02.30.Jr, 02.60.-x, 02.30.Nw, 46.15.-x, 47.11.-j, 02.70.-c

Email address: strain@math.berkeley.edu (John Strain)
URL: http://math.berkeley.edu/∼strain (John Strain)

1. Introduction

An algorithm is presented for the fast evaluation of the Fourier transform

$$\widehat{f}(\tau_k) = \sum_{i=1}^{N} \int_{\Delta_i} \exp(\mathrm{i}\tau_k^T \sigma) f_i(\sigma)\, \mathrm{d}\sigma, \qquad 1 \le k \le N, \tag{1}$$

at arbitrary points $\tau_k \in \mathbf{R}^D$. Here[1] $\mathrm{i} = \sqrt{-1}$, polynomial densities f_i are defined on simplices $\Delta_i \subset \mathbf{R}^D$, the ambient dimension $D \ge 1$, and the simplex dimension d satisfies $0 \le d \le D$. Each Δ_i is a d-dimensional simplex consisting of all convex combinations of $d+1$ given vertices $v_j \in \mathbf{R}^D$:

$$\Delta = \left\{ \sum_{j=0}^{d} \theta_j v_j \mid \theta_j \ge 0, \sum_{j=0}^{d} \theta_j = 1 \right\}. \tag{2}$$

The algorithm evaluates transform (1) to accuracy ϵ, in work $O(N \log N \log^{D+1} \epsilon)$, for arbitrary d and D. It employs a butterfly algorithm: Group source simplices Δ_i and target points τ_k into hierarchical tree structures, approximate the kernel $\exp(\mathrm{i}t^T s)$ by low-rank expansions, and transform expansions from source-local to target-local form. An efficient new dimensional recurrence evaluates source-local Fourier transforms of polynomials over simplices. The derivation, analysis and Fortran 77 implementation all operate with arbitrary ambient dimension $D \ge 1$, simplex dimension $0 \le d \le D$, and polynomial degree $\deg(f_i) = p \ge 0$. Despite this generality, the implementation runs orders of magnitude faster than direct evaluation, and compares favorably with the specialized Fast Fourier Transform (FFT).

Preliminaries on error bounds, translation lemmas, and hierarchical tree structures are reviewed in Section 2. Section 3 combines these tools to derive a pointwise ($d = 0$) butterfly algorithm and analyze error propagation through the algorithm. Section 4 generalizes the algorithm to polynomials f_i on simplices of dimension $d \ge 1$. An efficient dimensional recurrence computes exponential-polynomial moments in $d \ge 1$ by a combination of simplex quadrature and recurrence $d \to d-1$. Section 5 verifies efficiency and accuracy by numerical experiments. Section 6 explores acceleration with optimized basis functions, application of Galerkin matrices, and extensions to Laplace and Gauss transforms.

The existing literature [1, 2, 3, 4, 5, 6, 7, 8, 9] applies butterfly algorithms to various kernels. The data of these algorithms consists of point values at given locations on curves or surfaces in 1 through 3 dimensions. The approximation technique employed is usually Chebyshev interpolation [2, 3, 6, 7] and occasionally interpolative decomposition [4]. They obtain 12-digit accuracy in 1 dimension [4, 7], and 3 to 6 digit accuracy in 2 and 3 dimensions [2, 3, 6, 9].

We apply butterfly algorithms to the Fourier kernel. The data generalizes point values to polynomials on simplices in arbitrary dimension and codimension. The approximation technique employed is Taylor expansion, which simplifies the generalization to polynomials and obtains controllable 3 to 12 digit accuracy.

[1] Notation is summarized in Table 1.

Table 1: Notation

D	Ambient dimension of $\mathbf{R}^D$	d	Simplex dimension
f	Degree-p polynomial source density	$\hat{f}$	Fourier transform
p	Degree of source density	P	Number of coefficients $\binom{p+d}{d}$
σ	Source point $(\sigma_1, \ldots, \sigma_D)^T \in \mathbf{R}^D$	τ	Target point $(\tau_1, \ldots, \tau_D)^T \in \mathbf{R}^D$
Δ	Source simplex of dimension d	Λ	Target simplex of dimension d
N_S	Number of sources σ_i or Δ_i	N_T	Number of targets τ_i or Λ_i
$\mathcal{S}_L$	L-level source tree	$\mathcal{T}_L$	L-level target tree
S	Cubical source cell	T	Cubical target cell
s	Center $(s_1, \ldots, s_D)^T \in \mathbf{R}^D$ of S	t	Center $(t_1, \ldots, t_D)^T \in \mathbf{R}^D$ of T
$R(S)$	Radius of S	$R(T)$	Radius of T
ρ_S	Side length $R(S)/\sqrt{D}$	ρ_T	Side length $R(T)/\sqrt{D}$
m	Order of Taylor expansion	M	Length $\binom{m+D}{D}$ of Taylor expansion
ϵ	Accuracy $(Re/m)^m$	R	Accuracy radius $R(S)R(T)$
α	Multiindex $(\alpha_1, \ldots, \alpha_D) \in N^D$	$\lvert\alpha\rvert$	Order $\lvert\alpha_1\rvert + \cdots + \lvert\alpha_D\rvert$
$\alpha!$	Factorial $\alpha_1! \cdots \alpha_D!$	t^α	Monomial $t_1^{\alpha_1} \cdots t_D^{\alpha_D}$
$t^T s$	Inner product $t_1 s_1 + \cdots + t_D s_D$	$C_\alpha(S,T)$	Coefficients representing S to T
$A_{\alpha\beta}(s)$	Source translation matrix	$B_{\alpha\beta}(s,t)$	Target translation matrix
$F_{\alpha\beta}(s,t)$	Error propagation matrix	$\mu(R,m)$	Stability bound
$G_\alpha(d,\Delta,f,t)$	Exponential-polynomial moments	V	Simplex matrix $[v_1 - v_0] \cdots \lvert v_d - v_0]$
$\lvert\Delta\rvert$	Simplex volume $\sqrt{\det(V^T V)}$	H	Hyperplane containing simplex
$\partial\Delta$	Relative boundary $\cup_{j=0}^d \partial_j \Delta$	n	Relative outward unit normal
$t_\parallel$	Parallel component $V(V^T V)^{-1} V^T t$	$t_\perp$	Perpendicular component $t - t_\parallel$
E	Moment shift vector	z	Shift vector $-\mathrm{i}t_\parallel / \lVert t_\parallel \rVert^2$

Special cases of the piecewise-polynomial Fourier transform (1) in which the simplices are discrete points ($d = 0$), line segments ($d = 1$), triangles ($d = 2$) or tetrahedra ($d = 3$) occur frequently in applications.

Points: When the simplex dimension $d = 0$, each simplex Δ_i is a point $\sigma_i \in \mathbf{R}^D$, and each polynomial f_i is a constant complex number. The piecewise-polynomial Fourier transform reduces to the pointwise nonuniform Fourier transform

$$\widehat{f}(\tau_k) = \sum_{i=1}^{N} \exp(\mathrm{i}\tau_k^T \sigma_i) f_i, \qquad 1 \le k \le N. \tag{3}$$

If $\sigma_i = 2\pi i/N$ and $\tau_k = k$ are restricted to equidistant grids and $D = 1$, transform (3) reduces to the classical uniform FFT [10]. Several classes of fast algorithms for unrestricted σ_i and τ_k are well-known [11, 12, 13, 14, 15]. Physical applications of transform (3) include threshold dynamics [16], magnetic resonance imaging [17], and a host of others. Often these applications involve higher-dimensional objects which are discretized into pointwise transforms by quadrature formulas with dozens of points per wavelength.

Lines: When the simplex dimension $d = 1$, each simplex Δ_i is a line segment $[u_i, v_i]$ connecting endpoints u_i, $v_i \in \mathbf{R}^D$, and parametrized by $\sigma = u_i + \phi(v_i - u_i)$ for $0 \le \phi \le 1$. The piecewise-polynomial Fourier transform reads

$$\widehat{f}(\tau_k) = \sum_{i=1}^{N} \|v_i - u_i\| \exp(\mathrm{i}\tau_k^T u_i) \int_0^1 \exp(\mathrm{i}\phi\tau_k^T (v_i - u_i)) f_i(\phi)\, \mathrm{d}\phi \tag{4}$$

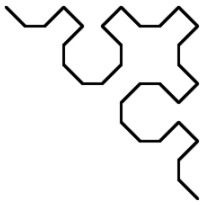
for $1 \le k \le N$. E.g. Fig. 1 displays two L-level approximate Sierpinski paths [18], composed of line segments Δ_i with simplex dimension $d = 1$ in ambient dimension $D = 2$, and the Fourier transform amplitudes $|\hat{f}(t)|$ of unit data $f = 1$ on these paths. As L increases, the Sierpinski paths fill a triangle Δ, and $\widehat{f}$ converges to the Fourier transform of the indicator function of Δ.

Triangles: When $d = 2$, each simplex Δ_i is a triangle with vertices u_i, v_i, $w_i \in \mathbf{R}^D$, parametrized by $\sigma = u_i + \phi_1(v_i - u_i) + \phi_2(w_i - u_i)$ where $\phi_1 \ge 0$, $\phi_2 \ge 0$, $\phi_1 + \phi_2 \le 1$. The piecewise-polynomial Fourier transform reads

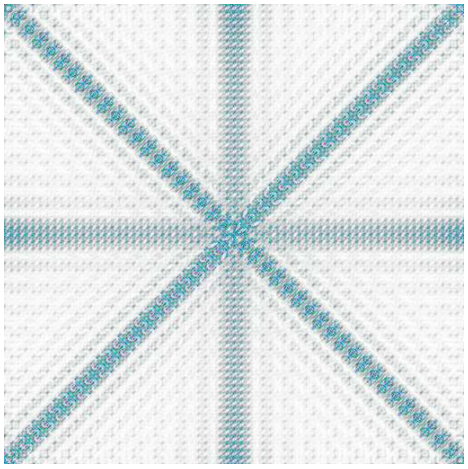
$$\begin{aligned} \widehat{f}(\tau_k) \quad = \quad & \sum_{i=1}^{N} \sqrt{\|v_i - u_i\|^2 \|w_i - u_i\|^2 - ((v_i - u_i)^T (w_i - u_i))^2} \\ & \exp(\mathrm{i}\tau_k^T u_i) \int_0^1 \exp(\mathrm{i}\phi_2 \tau_k^T (w_i - u_i)) \\ & \int_0^{1-\phi_2} \exp(\mathrm{i}\phi_1 \tau_k^T (v_i - u_i)) f_i(\phi_1, \phi_2)\, \mathrm{d}\phi_1\, \mathrm{d}\phi_2 \end{aligned} \tag{5}$$

for $1 \le k \le N$. Algorithms for evaluating transform (5) when each f_i is a polynomial are discussed in [19, 20, 21]. Physical applications of this transform include the solution of linear constant-coefficient $p \times q$ elliptic systems

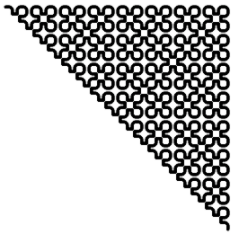
$$Au(x) = \sum_{j=1}^{D} A_j \partial_j u(x) + A_0 u(x) = f(x),$$



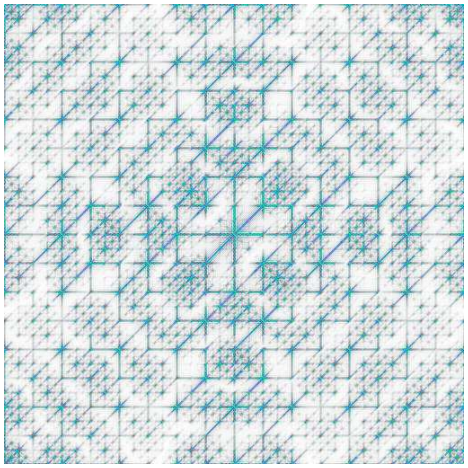
(a) 5-level path



(b) 8-fold symmetry



(c) 10-level path



(d) 3-fold symmetry

Figure 1: Sierpinski paths and Fourier transform amplitudes

in a domain $\Omega \subset \mathbf{R}^D$, with $r \times q$ full-rank boundary conditions $B(\gamma)u(\gamma) = g(\gamma)$ imposed on the boundary $\Gamma = \partial\Omega$. Such systems include Laplace, Helmholtz, Maxwell and Stokes equations, and can be efficiently solved by integral equation methods based on Ewald summation [22]. These methods rely on the $p \times q$ matrix-valued Fourier transform

$$\int_\Gamma A_\nu(\sigma) P(\sigma) \mathrm{e}^{-\mathrm{i}\tau^T \sigma} \mathrm{d}\,\sigma$$

where $A_\nu(\sigma) = \sum_j \nu_j(\sigma) A_j$, $P(\sigma) = I - B(\sigma)^*(B(\sigma)B(\sigma)^*)^{-1}B(\sigma)$, and ν is the outward unit normal to Γ. These transforms are usually discretized by quadrature formulas employing dozens of points per wavelength, making a piecewise-polynomial algorithm an attractive alternative.

2. Preliminaries

We present butterfly algorithms as a combination of low-rank approximation and hierarchical point clustering, via the following techniques:

- The exponential kernel $\exp(\mathrm{i}t^T s)$ of Fourier transform (1) is approximated by low-rank expansion.
- A simple error bound delineates regions where expansion is accurate.
- If sources and targets are all close together, a single global expansion separates the variables σ and τ. In general, many local expansions are needed, each representing the effect of a cluster of sources on a cluster of targets.
- Translation lemmas split and merge expansions between different target and source clusters.
- Hierarchical tree structures organize source and target points into clusters where low-rank expansion is accurate.

Pointwise algorithms are constructed from these preliminaries in Section 3. The piecewise-polynomial algorithm of Section 4 brings in two additional techniques: dimensional recurrence and approximate clustering.

2.1. Low-rank Taylor expansion

The multidimensional Taylor expansion

$$\exp(\mathrm{i}t^T s) = \sum_{n=0}^{\infty} \frac{\mathrm{i}^n \left(\sum_{j=1}^{D} t_j s_j\right)^n}{n!} = \sum_{\alpha \geq 0} \frac{\mathrm{i}^{|\alpha|}}{\alpha!} t^\alpha s^\alpha \tag{6}$$

of the complex exponential kernel follows immediately from the multinomial theorem

$$\left(\sum_{j=1}^{D} t_j\right)^n = n! \sum_{|\alpha|=n} \frac{t^\alpha}{\alpha!}. \tag{7}$$

2.2. Error bound

Let E_m be the error incurred by terminating expansion (6) after $M = \binom{m+D}{D}$ terms of order $|\alpha| \leq m$. Stirling's inequality $m! \geq (m/\mathrm{e})^m$ implies that[23]

$$|E_m| = \left| \sum_{|\alpha|>m} \frac{\mathrm{i}^{|\alpha|}}{\alpha!} t^\alpha s^\alpha \right| = \left| \sum_{k=m+1}^{\infty} \frac{\mathrm{i}^k}{k!} (t^T s)^k \right| \leq \sum_{k=m+1}^{\infty} \frac{R^k}{k!} \leq \left(\frac{R\mathrm{e}}{m}\right)^m, \tag{8}$$

whenever $|t^T s| \leq R$ and $m \geq 3.8R$. Table 2 lists m and M which guarantee accuracy ϵ: E.g. with $R = 1$ the error bound (8) guarantees accuracy $\epsilon = 10^{-6}$ with $m = 11$ and accuracy $\epsilon = 10^{-12}$ with $m = 16$.

2.3. Separation of variables

A thought experiment involving unrealistically placed targets and sources demonstrates the utility of low-rank expansion. Suppose all the sources σ_i and targets τ_k in the pointwise transform (3) satisfy $|\tau_k^T \sigma_i| \le R$, where R and m are chosen to guarantee error $|E_m| \le \epsilon$. Then expansion (6) separates the variables τ and σ to speed up the pointwise transform:

$$\widehat{f}(\tau_k) = \sum_{i=1}^{N} \exp(\mathrm{i}\tau_k^T \sigma_i) f_i = \sum_{i=1}^{N} \sum_{|\alpha| \le m} \frac{\mathrm{i}^{|\alpha|}}{\alpha!} \tau_k^\alpha \sigma_i^\alpha f_i + F_m = \sum_{|\alpha| \le m} C_\alpha \tau_k^\alpha + F_m.$$

Here the M coefficients C_α, defined by

$$C_\alpha = \frac{\mathrm{i}^{|\alpha|}}{\alpha!} \sum_{i=1}^{N} \sigma_i^\alpha f_i,$$

encode the sources σ_i and strengths f_i, and $|F_m| \le \epsilon \sum |f_i|$ bounds the error. $O(MN)$ work suffices to compute coefficients C_α for $|\alpha| \le m$ and evaluate $\widehat{f}(\tau_k)$ for $1 \le k \le N$. Thus the rank-M kernel approximation

$$\exp(\mathrm{i}t^T s) = \sum_{|\alpha| \le m} \frac{\mathrm{i}^{|\alpha|}}{\alpha!} t^\alpha s^\alpha + E_m \tag{9}$$

gives an $O(N)$ algorithm, with a constant factor $O(M)$ depending polylogarithmically on the accuracy ϵ.

2.4. Local expansions

The sources σ_i and targets τ_k are not conveniently clustered in most applications (e.g. the classical uniform FFT where $0 \le \tau_k \sigma_i \le 2\pi N \to \infty$ as $N \to \infty$). Thus we employ a collection of low-rank expansions. Each expansion represents the Fourier transform of sources σ_i in a cubical cell S centered at s with radius $R(S)$, evaluated at targets τ_k in a cell T centered at t with radius $R(T)$:

$$\begin{aligned}
\sum_{\sigma_i \in S} \exp(\mathrm{i}\tau_k^T \sigma_i) f_i &= \sum_{\sigma_i \in S} \exp(\mathrm{i}\tau_k^T s + \mathrm{i}(\tau_k - t)^T (\sigma_i - s) + \mathrm{i}t^T \sigma_i - \mathrm{i}t^T s) f_i \\
&= \exp(\mathrm{i}t^T s) \sum_{|\alpha| \le m} \frac{\mathrm{i}^{|\alpha|}}{\alpha!} \sum_{\sigma_i \in S} (\sigma_i - s)^\alpha \exp(\mathrm{i}t^T (\sigma_i - s)) f_i \\
&\quad (\tau_k - t)^\alpha \exp(\mathrm{i}(\tau_k - t)^T s) + F_m \\
&= \sum_{|\alpha| \le m} C_\alpha(S,T) \left(\frac{\tau_k - t}{R(T)} \right)^\alpha \exp(\mathrm{i}(\tau_k - t)^T s) + F_m.
\end{aligned} \tag{10}$$

The coefficients $C_\alpha(S,T)$ are given by

$$C_\alpha(S,T) = \exp(\mathrm{i}t^T s) \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \sum_{\sigma_i \in S} \left(\frac{\sigma_i - s}{R(S)} \right)^\alpha \exp(\mathrm{i}t^T (\sigma_i - s)) f_i \tag{11}$$

where $R = R(S)R(T)$. The error $|F_m| \leq \epsilon \sum |f_i|$ since all $\tau_k \in T$ and $\sigma_i \in S$ satisfy

$$|(\tau_k - t)^T(\sigma_i - s)| \leq R(S)R(T) = R \quad \text{where} \quad \left(\frac{Re}{m}\right)^m \leq \epsilon. \tag{12}$$

Thus the Fourier transform (10) of sources in S, evaluated at targets in T, is approximated with a kernel

$$E_{TS}(\tau, \sigma) = \exp(-\mathrm{i}t^T s) \exp(\mathrm{i}t^T \sigma) \exp(\mathrm{i}\tau^T s) \sum_{|\alpha| \leq m} \left(\frac{\sigma - s}{R(S)}\right)^\alpha \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \left(\frac{\tau - t}{R(T)}\right)^\alpha \tag{13}$$

of rank at most $M = \binom{m+D}{D}$. The geometry of S and T control the accuracy of this approximation via inequality (12).

2.5. Translation lemmas

We apply the low-rank approximate kernel (13) to hierarchical clusters of sources and targets, via a pair of lemmas that translate the coefficients $C_\alpha(S, T)$ of expansions (10), to smaller target cells and larger source cells. Lemma 1 follows from expansion (6), while the error bound follows from inequality (12). Lemma 2 follows from the multinomial theorem (7).

Lemma 1. (Smaller target cells) *Suppose coefficients $C(S_1, T_0)$ represent sources $\sigma_i \in S_1$ to targets $\tau \in T_0$ with $R(S_1)R(T_0) = R$, and a smaller target cell $T_1 \subset T_0$ has center t_1 and radius $R(T_1) = R(T_0)/2$. Then the coefficients*

$$C_\alpha(S_1, T_1) = \exp(\mathrm{i}t_1^T s_1) \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \sum_{\sigma_i \in S_1} \left(\frac{\sigma_i - s_1}{R(S_1)}\right)^\alpha \exp(\mathrm{i}t_1^T(\sigma_i - s_1)) f_i$$

are given by an upper triangular matrix multiply

$$\begin{aligned} C_\alpha(S_1, T_1) &= \exp(\mathrm{i}(t_1 - t_0)^T s_1) \sum_{\beta \geq 0} \binom{\beta + \alpha}{\beta} 2^{-|\beta|} \left(\frac{t_1 - t_0}{R(T_1)}\right)^\beta C_{\beta+\alpha}(S_1, T_0) \\ &= \sum_{\beta \geq \alpha} B_{\alpha\beta}(s_1, t_1 - t_0) C_\beta(S_1, T_0). \end{aligned} \tag{14}$$

The matrix elements $B_{\alpha\beta}$ are given by

$$B_{\alpha\beta}(s, t) = \exp(\mathrm{i}(t^T s)) \binom{\beta}{\alpha} 2^{|\alpha| - |\beta|} \left(\frac{t}{R(T_1)}\right)^{\beta - \alpha} \quad \text{for} \quad \beta \geq \alpha. \tag{15}$$

If $|\alpha| \leq m$ and $|(t_1 - t_0)^T(\sigma_i - s_1)| \leq R/\rho$, then the error E incurred by truncating formula (14) after terms of order $|\beta| \leq m$ is bounded by $\rho^{-m} \epsilon \sum |f_i|$.

Lemma 2. (Larger source cells) *Suppose coefficients $C(S_1, T_1)$ centered at s_1 represent sources $\sigma_i \in S_1$ to targets $\tau \in T_1$, and a larger source cell $S_0 \supset S_1$ has center s_0 and radius $R(S_0) = 2R(S_1)$ where $R(S_0)R(T_1) = R$. Let*

$$C_\alpha^1(S_0, T_1) = \exp(\mathrm{i}t_1^T s_0) \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \sum_{\sigma_i \in S_1} \left(\frac{\sigma_i - s_0}{R(S_0)}\right)^\alpha \exp(\mathrm{i}t_1^T(\sigma_i - s_0)) f_i,$$

be the coefficients centered at s_0 which represent sources $\sigma_i \in S_1 \subset S_0$ to targets $\tau \in T_1$. Then

$$C_\alpha^1(S_0, T_1) = \sum_{\beta \le \alpha} A_{\alpha\beta}(s_1 - s_0) C_\beta(S_1, T_1) \tag{16}$$

where the matrix elements $A_{\alpha\beta}$ are given by

$$A_{\alpha\beta}(s) = 2^{-|\alpha|} \frac{(\mathrm{i}R)^{|\alpha-\beta|}}{(\alpha-\beta)!} \left(\frac{s}{R(S_1)}\right)^{\alpha-\beta} \quad \textit{for} \quad \beta \le \alpha. \tag{17}$$

Usually a larger source cell S_0 is the union of $n = 2^D$ subcells S_{1j}, with coefficients $C(S_{1j}, T_1)$ representing sources in S_{1j} to targets $\tau \in T_1$. After Lemma 2 applies matrix $A(s_{1j} - s_0)$ to shift each coefficient vector to the common center s_0, all the sources in S_0 are represented to T_1 by a single coefficient vector

$$C_\alpha(S_0, T_1) = \sum_{j=0}^{n-1} \sum_{\beta \le \alpha} A_{\alpha\beta}(s_{1j} - s_0) C_\beta(S_{1j}, T_1).$$

2.6. Hierarchical point clustering

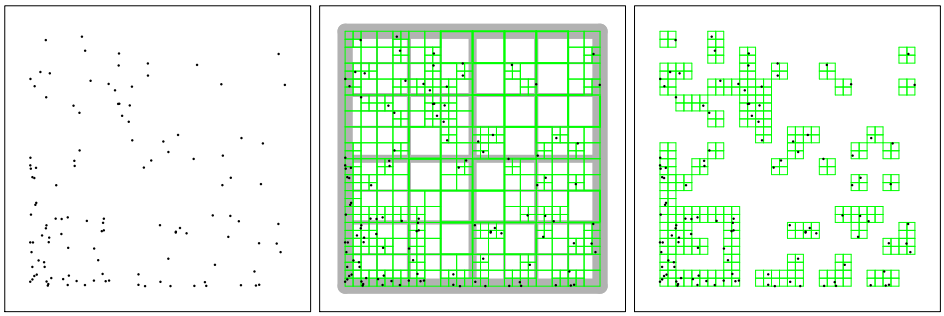
Translation lemmas work well within a data structure which clusters sources and targets σ_i and τ_k into cells S and T satisfying inequality (12). Collections of geometric objects, such as the source simplices and target points in the Fourier transform (1), are efficiently clustered into local cells by a 2^D-ary tree [24]. Algorithm 1 constructs a tree $\mathcal{S}_L$ which organizes points σ_i (Fig. 2). The algorithm is modified for simplices ($d > 0$) in Section 4.2.

Table 2: Order m and number M of coefficients for 3, 6, 9 and 12-digit accuracy in dimensions $D = 1$ through 3.

R	ϵ	$\lceil \lvert \log(\epsilon) \rvert \rceil$	m	$M = \binom{m+1}{1}$	$M = \binom{m+2}{2}$	$M = \binom{m+3}{3}$
$2/\pi$	10^{-3}	7	6	7	28	84
	10^{-6}	14	9	10	55	220
	10^{-9}	21	12	13	91	455
	10^{-12}	28	14	15	120	680
1	10^{-3}	7	8	9	45	165
	10^{-6}	14	11	12	78	364
	10^{-9}	21	14	15	120	680
	10^{-12}	28	16	17	153	969
$\pi/2$	10^{-3}	7	10	11	66	286
	10^{-6}	14	13	14	105	560
	10^{-9}	21	16	17	153	969
	10^{-12}	28	19	20	210	1540

Algorithm 1 Hierarchical point clustering by a 2^D-ary tree, $Q = [-1, 1]^D$.

Create level-0 root cell $S_{00} = s_{00} + \rho Q$ containing all points σ_i
Make pointers $\sigma_i \leftrightarrow S_{00}$ for all i
for $l = 1 \dots L - 1$
  for $I = 0 \dots 2^{Dl} - 1$
    $S_{lI} = s_{lI} + 2^{-l} \rho Q$
    for $j = 0 \dots 2^D - 1$
    Create level-l child cell $S_{l+1,2^D I+j} = s_{l+1,2^D I+j} + 2^{-l-1} \rho Q \subset S_{lI}$
    Make pointers $\sigma_i \leftrightarrow S_{l+1,2^D I+j}$ for $\sigma_i \in S_{l+1,2^D I+j}$



(a) Points (b) Tree (c) Cells

Figure 2: Points in two dimensions, a 6-level tree, and nonempty leaf cells.

3. A pointwise butterfly algorithm

The pointwise Fourier transform (3) can be efficiently approximated by low-rank expansion and hierarchical point clustering [1, 2, 3, 4, 5, 6, 7, 8, 9]. We derive these approximations in four steps:

- localize sources and targets into hierarchical tree structures,
- compute source-local coefficients,
- split and merge coefficients systematically in a butterfly scheme,
- evaluate target-local expansions.

Since we employ Taylor expansion in place of the usual Chebyshev expansion or interpolative decomposition, the derivation and accompanying error analysis (Section 3.5) generalize to the piecewise-polynomial Fourier transform (1) in Section 4.

3.1. Localization

Hierarchical tree structures $\mathcal{S}_L$ and $\mathcal{T}_L$ cover the source and target regions S and T by superimposed levels from root $l = 0$ to leaves $l = L$ (Fig. 3). Let $R(S)$ and $R(T)$ be the radii of S and T. On level l, the sources σ_i and targets τ_k are localized into 2^{Dl} cubical cells S_{lI} or T_{lK} with radii $R(S_{lI}) = 2^{-l}R(S)$ or $R(T_{lK}) = 2^{-l}R(T)$, satisfying

$$S = \bigcup_{I=0}^{2^{Dl}-1} S_{lI}, \qquad T = \bigcup_{K=0}^{2^{Dl}-1} T_{lK}.$$

Suppose the number of levels L and the order m of Taylor expansion satisfy

$$2^{-l}R(S)\,2^{l-L}R(T) \le R \qquad \text{and} \qquad \left(\frac{R\mathrm{e}}{m}\right)^m \le \epsilon. \tag{18}$$

Then for sources σ_i in a level-l source cell S_{lI} and targets τ_k in a level-$(L-l)$ target cell $T_{L-l,K}$ (Fig. 3), the low-rank approximate kernel (13) will be accurate to $\epsilon \sum |f_i|$.

3.2. Source-local coefficients

For $I = 0$ to $2^{DL} - 1$, source-local coefficients

$$C_\alpha(S_{LI}, T_{00}) = \exp(\mathrm{i}t_{00}^T s_{LI}) \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \sum_{\sigma_i \in S_{LI}} \left(\frac{\sigma_i - s_{LI}}{R(S_{LI})}\right)^\alpha \exp(\mathrm{i}t_{00}^T(\sigma_i - s_{LI}))f_i$$

represent sources σ_i in each source leaf cell S_{LI}, to targets τ_k in the target root cell T_{00} (Fig. 4). Each source σ_i contributes to M coefficients in a single leaf cell S_{LI}. Thus the total cost of computing source-local coefficients is $O(NM)$. Given source-local coefficients, the pointwise Fourier transform (3) could be expensively approximated at each target point τ_k by summing the 2^{DL} source-local expansions:

$$\widehat{f}(\tau_k) = \sum_{I=0}^{2^{DL}-1} \left[\exp(\mathrm{i}(\tau_k - t_{00})^T s_{LI}) \sum_{|\alpha| \le m} C_\alpha(S_{LI}, T_{00}) \left(\frac{\tau_k - t_{00}}{R(T_{00})}\right)^\alpha + F_m \right].$$

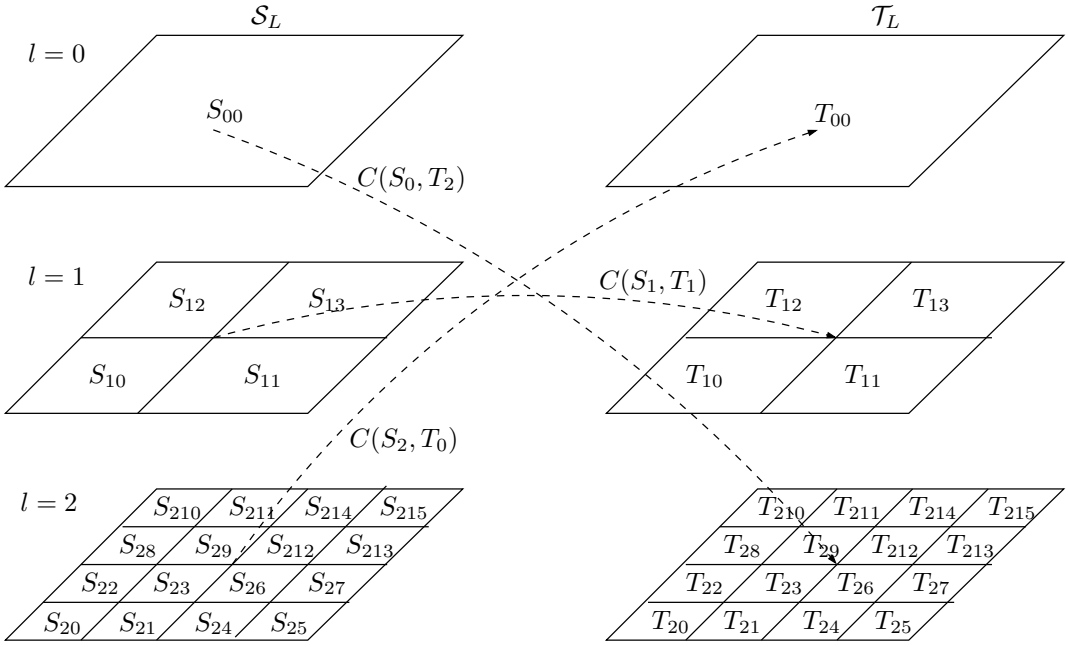


Figure 3: Expansion coefficients $C(S_l, T_{L-l})$ accurately represent opposite levels l and $L-l$ of the source and target trees $\mathcal{S}_L$ and $\mathcal{T}_L$. Here $D = L = 2$.

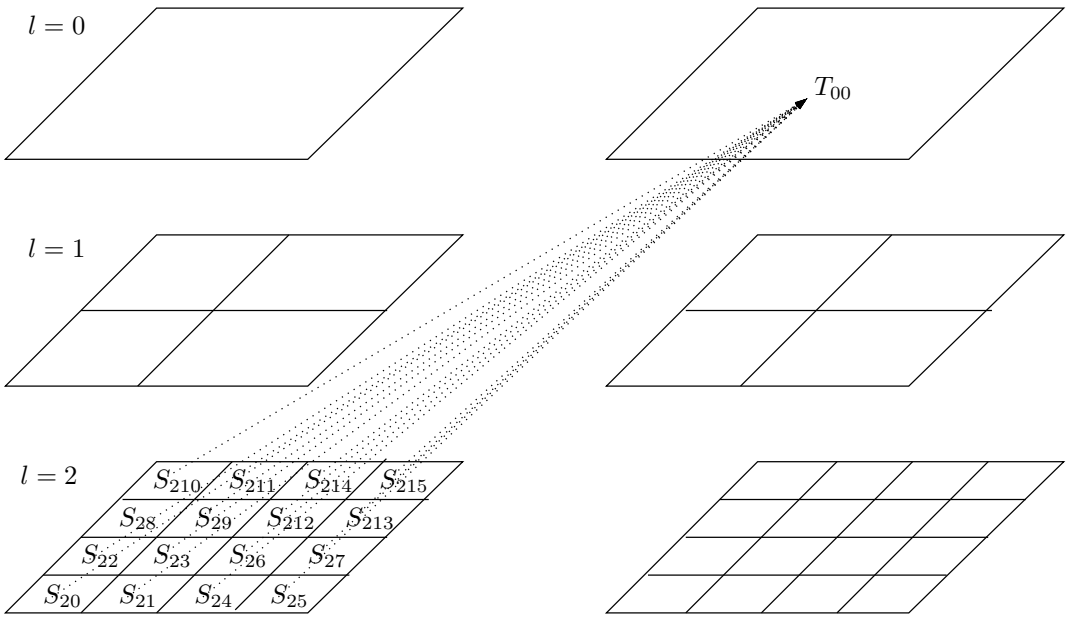


Figure 4: $2^{DL} = 16$ source-local coefficient vectors $C(S_{2I}, T_{00})$ represent sources in each source leaf cell S_{2I} to all targets.

3.3. Butterfly scheme

The butterfly scheme recursively converts source-local coefficients $C(S_L, T_0)$ to target-local coefficients $C(S_0, T_L)$, which represent all the sources σ_i to targets τ_k in each target leaf cell T_{LK}:

$$C(S_L, T_0) \to C(S_{L-1}, T_1) \to C(S_{L-2}, T_2) \to \cdots \to C(S_1, T_{L-1}) \to C(S_0, T_L).$$

For $l = L$ to 1, each step of the scheme converts 2^{DL} coefficients for source level l and target level $L - l$, to source parent level $l - 1$ and target child level $L - l + 1$, via *split* and *merge* operations. Each step of *split* and *merge* halves the target cells, doubles the source cells, and preserves the error bound (12) (Fig. 5).

3.3.1. Split

On source level l and target level $L-l$, each of $2^{D(L-l)}$ target cells $T_{L-l,K}$ *splits* into 2^D children $T_{L-l+1,2^D K+k} \subset T_{L-l,K}$. Each coefficient vector $C(S_{lI}, T_{L-l,K})$ yields 2^D child coefficient vectors

$$C(S_{lI}, T_{L-l+1,2^D K+k}) = B(s_{lI}, t_{L-l+1,2^D K+k} - t_{L-l,K}) C(S_{lI}, T_{L-l,K}),$$

via the $M \times M$ matrices $B(s, t)$ defined in Eq. (15). The error in each child coefficient vector is bounded by $2^{-m} \epsilon \sum_{|} f_i|$, according to Lemma 1.

3.3.2. Merge

Each group of 2^D level-l source cell siblings $S_{l,2^D I+i} \subset S_{l-1,I}$, *merges* into their parent $S_{l-1,I}$. Fix a level $L - l + 1$ target child cell $T_1 = T_{L-l+1,2^D K+k} \subset T_{L-l,K}$ and let $J = 2^D I$. Then each of $2^{D(L-l+1)}$ source child coefficient vectors $C(S_{l,J+i}, T_1)$ yields a partial parent coefficient vector

$$C^i(S_{l-1,I}, T_1) = A(s_{l,J+i} - s_{l-1,I}) C(S_{l,J+i}, T_1),$$

via the $M \times M$ matrices $A(s)$ defined in Eq. (17). Since the 2^D vectors C^i have the same source center $s = s_{l-1,I}$, and the same target center t_1, the total source parent coefficient vector is the sum

$$\begin{aligned} C(S, T_1) &= \sum_{i=1}^{2^D} C^i(S, T_1) \\ &= \sum_{i=1}^{2^D} A(s_{l,J+i} - s) B(s_{l,J+i}, t_1 - t_{L-l,K}) C(S_{l,J+i}, T_{L-l,K}). \end{aligned}$$

3.4. Target-local expansions

L steps of the butterfly scheme yield coefficient vectors $C(S_{00}, T_{LK})$ for target leaf cells T_{LK}. Each coefficient vector represents all the sources $\sigma_i \in S_{00}$ to any $\tau_k \in T_{LK}$ (Fig. 6):

$$\widehat{f}(\tau_k) = \exp\left(\mathrm{i}(\tau_k - t_{LK})^T s_{00}\right) \sum_{|\alpha| \le m} C_\alpha(S_{00}, T_{LK}) \left(\frac{\tau_k - t_{LK}}{R(T_{LK})}\right)^\alpha. \tag{19}$$

Thus evaluating the pointwise Fourier transform (3) at all targets τ_k in leaf cells T_{LK} costs $O(MN)$ work (Algorithm 2).

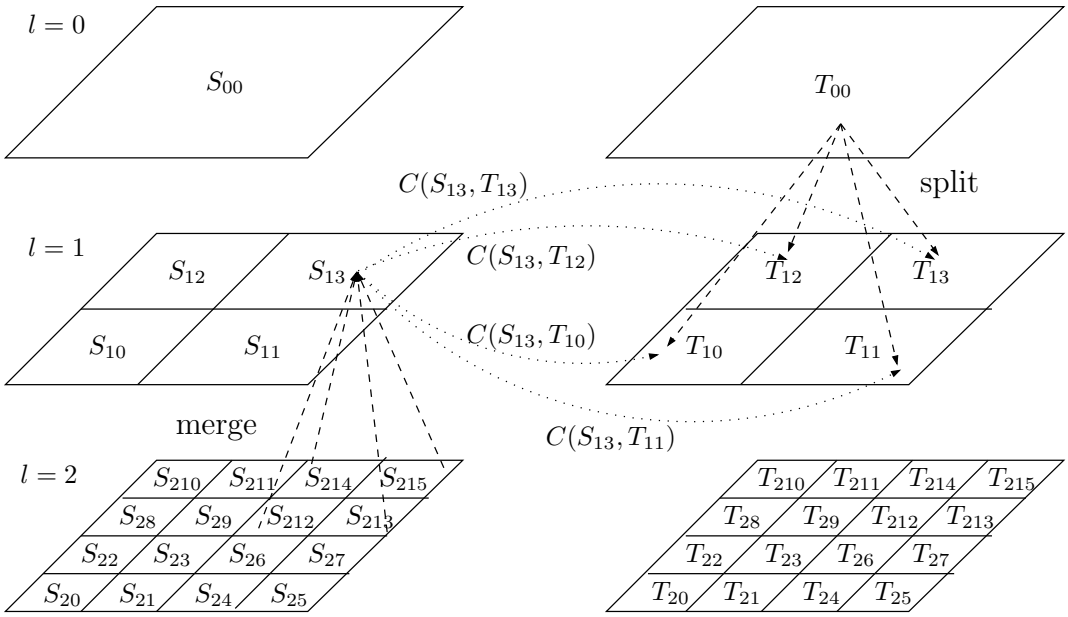


Figure 5: A single split and merge substep converts $2^D = 4$ level-2 source leaf/target root coefficient vectors $C(S_{2,12+i}, T_{00})$ to 4 source parent/target child coefficient vectors $C(S_{13}, T_{1K})$.

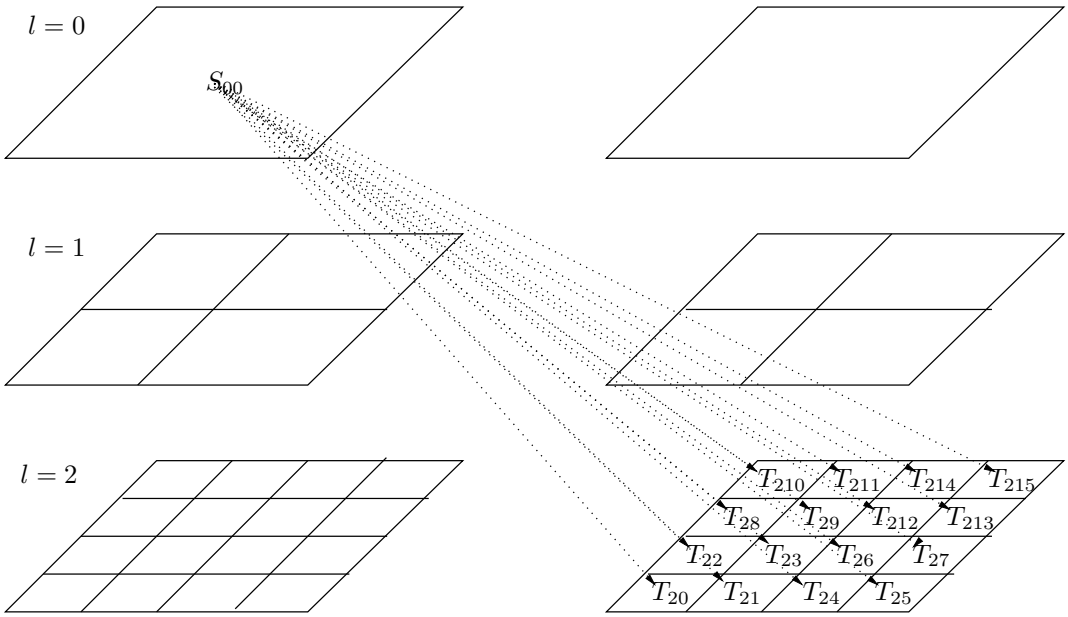


Figure 6: $2^{DL} = 16$ target-local coefficient vectors $C(S_{00}, T_{2K})$ represent all sources to each target leaf cell T_{2K}.

Algorithm 2 A pointwise butterfly algorithm

Step 1 - Localization

Sort sources σ_i and targets τ_k into leaf cells of L-level trees $\mathcal{S}_L$ and $\mathcal{T}_L$

Step 2 - Compute source-local coefficients

for $I = 0 \dots 2^{DL} - 1$
  $S = S_{LI}$
  $s = s_{LI}$
  $T = T_{00}$
  $t = t_{00}$
  for $|\alpha| \leq m$
    $C_\alpha(S,T) = \exp(\mathrm{i}t^T s)\frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!}\sum_{\sigma_i \in S}\left(\frac{\sigma_i - s}{R(S)}\right)^\alpha \exp(\mathrm{i}t^T(\sigma_i - s))f_i$

Step 3 - Butterfly scheme

for $l = L \dots 1$
  for $I = 0 \dots 2^{D(l-1)} - 1$
    $S = S_{l-1,I}$
    $s = s_{l-1,I}$
    $J = 2^D I$
    for $K = 0 \dots 2^{D(L-l)} - 1$
      $T = T_{L-l,K}$
      $t = t_{L-l,K}$
      for $k = 0 \dots 2^D - 1$
        $T_1 = T_{L-l+1,2^D K+k}$
        $t_1 = t_{L-l+1,2^D K+k}$
        $C(S,T_1) = \sum_{i=0}^{2^D-1} A(s_{l,J+i} - s)B(s_{l,J+i}, t_1 - t)C(S_{l,J+i}, T)$

Step 4 - Evaluate target-local expansions

for $K = 0 \dots 2^{DL} - 1$
  $S = S_{00}$
  $s = s_{00}$
  $T = T_{LK}$
  $t = t_{LK}$
  for $\tau_k \in T$
    $\widehat{f}(\tau_k) = \exp(\mathrm{i}(\tau_k - t)^T s)\sum_{|\alpha| \leq m} C_\alpha(S,T)\left(\frac{\tau_k - t}{R(T)}\right)^\alpha$

3.5. Error analysis

We validate our algorithm, showing that each coefficient truncation error grows by a factor $\mu \leq 10$ as it propagates through $L \leq 10$ butterfly steps. Previous butterfly algorithms [1, 2, 3, 4, 5, 6, 7, 8, 9] have usually relied on the spectral accuracy of recursive Chebyshev interpolation or interpolative decomposition to bound error propagation. For Taylor expansion of the Fourier kernel, explicit formulas for merging and shifting operators simplify the analysis considerably.

Error propagates from one leaf cell S_L at a time. Exact computation would transform the infinite source-local coefficient vector $C(S_L, T_0)$ to an infinite target-local coefficient vector $C(S_0, T_L)$ for each target leaf cell T_L:

$$C(S_L, T_0) \to C(S_{L-1}, T_1) \to \cdots \to C(S_1, T_{L-1}) \to C(S_0, T_L). \tag{20}$$

Each arrow $C(S_1, T_0) \to C(S_0, T_1) = EC(S_1, T_0)$ is implemented by an infinite matrix E with elements

$$\begin{aligned} E_{\alpha\beta} &= \sum_{\gamma \leq \min(\alpha,\beta)} A_{\alpha\gamma}(s_1 - s_0) B_{\gamma\beta}(s_1, t_1 - t_0) \\ &= \exp(i(t_1 - t_0)^T s_1) \sum_{\gamma \leq \min(\alpha,\beta)} \frac{(\mathrm{i}R/2)^{|\alpha|-|\gamma|}}{(\alpha-\gamma)!} 2^{-|\beta|} \binom{\beta}{\gamma} \left(\frac{s_1 - s_0}{R(S_1)}\right)^{\alpha-\gamma} \left(\frac{t_1 - t_0}{R(T_1)}\right)^{\beta-\gamma}. \end{aligned}$$

Since s_0 is a corner point of S_1 and t_0 is a corner point of T_1,

$$\frac{s_0 - s_1}{R(S_1)} = (\pm 1, \ldots, \pm 1) \qquad \text{and} \qquad \frac{t_1 - t_0}{R(T_1)} = (\pm 1, \ldots, \pm 1).$$

Thus there is a set $\mathcal{F}$ of 2^{2D} different matrices F with elements

$$F_{\alpha\beta} = \sum_{\gamma \leq \min(\alpha,\beta)} \frac{(\mathrm{i}R/2)^{|\alpha|-|\gamma|}}{(\alpha-\gamma)!} 2^{-|\beta|} \binom{\beta}{\gamma} (\pm 1, \ldots, \pm 1)^{\alpha-\gamma} (\pm 1, \ldots, \pm 1)^{\beta-\gamma},$$

which is independent of all numerical parameters except R. Each arrow in Eq. (20) is implemented by a scalar unitary prefactor $\exp(i(t_1 - t_0)^T s_1)$ and a matrix $F \in \mathcal{F}$ determined by the positions of the child cells S_1 and T_1 relative to their parents.

Accordingly, exact computation produces

$$C(S_0, T_L) = \omega F_L F_{L-1} \cdots F_1 C(S_L, T_0),$$

while the truncated computation produces the approximate value

$$\widehat{C}(S_0, T_L) = \omega P_m F_L P_m F_{L-1} P_m \cdots F_1 P_m C(S_L, T_0) = \left(\prod_{j=0}^{L-1} P_m F_{L-j} P_m \right) C(S_L, T_0).$$

Here $|\omega| = 1$, each $F_j \in \mathcal{F}$, and P_m projects onto the first M coefficients. Since

$$F_L F_{L-1} \cdots F_1 - P_m F_L P_m F_{L-1} \cdots F_1 P_m = \sum_{j=0}^{L} \left(\prod_{k=0}^{j-1} P_m F_{L-k} P_m \right) (I - P_m) \prod_{k=j}^{L-1} F_{L-k}$$

telescopes, the error analysis is completed by estimating the projection $I - P_m$ on exact coefficients $C(S_{L-j}, T_j) = F_j \cdots F_1 C(S_0, T_L)$, and bounding the truncated product.

Since scaled monomials satisfy

$$\left\| \frac{\sigma - s}{R(S)} \right\|_\infty \le 1 \qquad \Rightarrow \qquad \left| \left(\frac{\sigma - s}{R(S)} \right)^\alpha \right| \le 1,$$

the projection $I - P_m$ applied to any coefficient vector $C(S, T)$ gives

$$\|(I - P_m)C\|_2 = \left(\sum_{|\alpha| > m} |C_\alpha|^2 \right)^{1/2} \le \left(\frac{R\mathrm{e}}{m} \right)^m \sum_{\sigma_i \in S} |f_i|. \tag{21}$$

Numerical computations for $1 \le D \le 3$, $2/\pi \le R \le \pi/2$, $m = 2$ to 30 and $1 \le L \le 10$ show that (a) the stability bound[2]

$$\mu(R, m) = \max_{1 \le L \le 10} \max_{F_j \in \mathcal{F}} \left\| \prod_{k=1}^{L} P_m F_j P_m \right\|_2 \tag{22}$$

grows very slowly as L and $m \ge R\mathrm{e}$ increase (Fig. 7), and (b) the lower bound

$$\mu(R, m) \ge \max_{1 \le L \le 10} \max_{F_j \in \mathcal{F}} \|(P_m F_j P_m)^L\|_2 \tag{23}$$

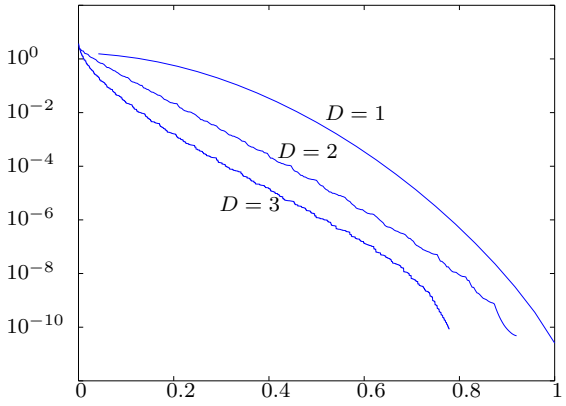
is sharp for $m > R\mathrm{e}$. For $m \le R\mathrm{e}$ the error bound $(R\mathrm{e}/m)^m$ is large so stability is irrelevant.

Combining Eqs. (21) and (22) bounds the error in a single source leaf cell due to truncating at each butterfly step by (Fig. 7)

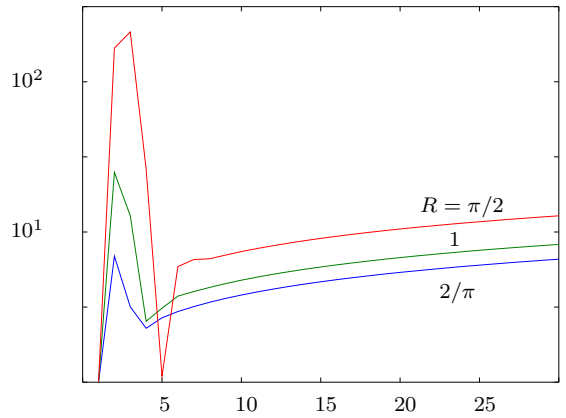
$$\|C(S_0, T_L) - \widehat{C}(S_0, T_L)\|_2 \le (L+1)\mu(R, m) \left(\frac{R\mathrm{e}}{m} \right)^m \sum_{\sigma_i \in S} |f_i|. \tag{24}$$

Since applying unitary prefactors and summing over source leaf cells enlarges the sum of source strengths f_i, the total error is bounded by $(L+1)\mu(R, m)\epsilon \sum_{i=1}^{N} |f_i|$. The numerical results of Section 5 support this analytical conclusion.

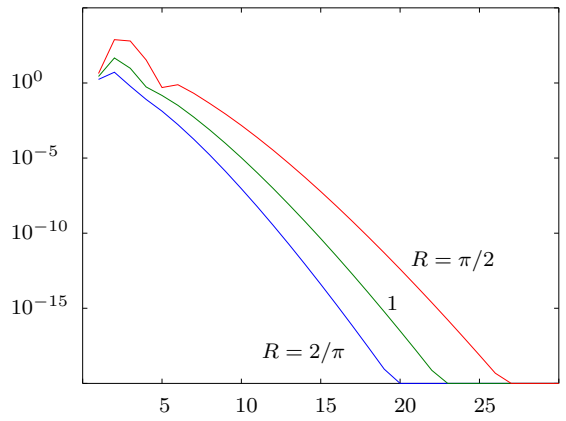
[2]Each $F \in \mathcal{F}$ has complex eigenvalues satisfying $\lambda_j(F) = 2^{1-j}\lambda_1(F)$ with $|\lambda_1(F)| = 1$, suggesting that products of these matrices should be bounded i.e. the joint spectral radius [25, 26] of $\mathcal{F}$ is 1.



(a) $\sigma_k(F_j)$ vs. k/M for $m = 24$ and $1 \leq D \leq 3$



(b) $\mu(R, m)$ vs. m



(c) $\mu(R, m)(\mathrm{Re}/m)^m$ vs. m

Figure 7: (a) Nonzero singular values of F_j, (b) stability constant, (c) and propagated error bound for $L \leq 10$.

4. A piecewise-polynomial butterfly algorithm

We enhance the pointwise butterfly algorithm (Algorithm 2) to evaluate the piecewise-polynomial Fourier transform

$$\widehat{f}(\tau_k) = \sum_{i=1}^{N} \int_{\Delta_i} \exp(\mathrm{i}\tau_k^T \sigma) f_i(\sigma)\,\mathrm{d}\sigma$$

with N points $\tau_k \in \mathbf{R}^D$ and N polynomial source densities f_i on d-dimensional source simplices $\Delta_i \subset \mathbf{R}^D$. The additional components are

- new dimensional recurrences for computing exponential-polynomial moments (Section 4.1), and
- a hierarchical tree structure for approximately localizing simplices with remainder (Section 4.2).

Suppose a source simplex Δ_i is contained in a cubical cell $S = s + \rho_S Q$ and target point $\tau_k \in T = t + \rho_T Q$, where $R = \sqrt{D}\rho_S\sqrt{D}\rho_T = R(S)R(T)$ satisfies

$$\left(\frac{R\mathrm{e}}{m}\right)^m \le \epsilon. \tag{25}$$

Then for $\sigma \in \Delta_i$,

$$\exp(\mathrm{i}\tau_k^T\sigma) = \sum_{|\alpha|\le m} \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \left(\frac{\sigma - s}{R(S)}\right)^{\alpha} \exp(\mathrm{i}t^T\sigma) \left(\frac{\tau_k - t}{R(T)}\right)^{\alpha} \exp(\mathrm{i}(\tau_k - t)^T s) + E_m$$

where $|E_m| \le \epsilon$. Hence integrating over Δ_i gives

$$\int_{\Delta_i} \exp(\mathrm{i}\tau_k^T\sigma) f_i(\sigma)\,\mathrm{d}\sigma = \sum_{|\alpha|\le m} \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} G_\alpha(d, \Delta_i, f_i, t) \left(\frac{\tau_k - t}{R(T)}\right)^{\alpha} \exp(\mathrm{i}(\tau_k - t)^T s) + F_m \tag{26}$$

where $|F_m| \le \epsilon \int |f_i|$. We define the M-vector G of exponential-polynomial moments of polynomial f on d-dimensional simplex Δ by

$$G_\alpha(d, \Delta, f, t) = \int_\Delta \left(\frac{\sigma - s}{R(S)}\right)^{\alpha} \exp(\mathrm{i}t^T\sigma) f(\sigma)\,\mathrm{d}\sigma. \tag{27}$$

The source center s and ambient dimension D are omitted from G for simplicity.

4.1. Exponential-polynomial moments G_α

The following recursive procedure computes exponential-polynomial moments G_α for all multiindices α with $|\alpha| \le m$:

1. If $d = 0$, compute the pointwise moments directly (Section 3.2).
2. If $0 < d < D$, extract the perpendicular variation (Section 4.1.1).
3. If the parallel variation is small, compute the moments by numerical quadrature (Section 4.1.2).
4. Otherwise, reduce the simplex dimension d to $d - 1$ by a dimensional recurrence (Section 4.1.3).

4.1.1. Extracting the perpendicular variation

If the simplex Δ has positive dimension and codimension ($0 < d < D$), then moments (27) are simplified by extracting the part of the target vector t which is perpendicular to Δ. Let the $D \times d$ full-rank matrix V have columns $v_i - v_0 \in \mathbf{R}^D$. Parametrize the simplex Δ by $\sigma = v_0 + V\theta$, where θ varies over the standard d-dimensional simplex

$$\Delta_0 = \left\{ (\theta_1, \ldots, \theta_d) \,|\, \theta_i \geq 0, \sum_{i=1}^{d} \theta_i \leq 1 \right\}.$$

Then the volume of Δ is $|\Delta| = \sqrt{\det(V^T V)}$ [27] and the affine hyperplane H containing Δ is

$$H = \left\{ v_0 + V\theta \,|\, \theta \in R^d \right\}.$$

Exponential-polynomial moments (27) become

$$\begin{aligned} G_\alpha(d, \Delta, f, t) &= |\Delta| \int_{\Delta_0} \exp(\mathrm{i}t^T(v_0 + V\theta)) \left(\frac{v_0 + V\theta - s}{R(S)} \right)^\alpha f(v_0 + V\theta)\, \mathrm{d}\theta \\ &= |\Delta| \exp(\mathrm{i}t_\perp^T v_0) \int_{\Delta_0} \exp(\mathrm{i}t_\parallel^T (v_0 + V\theta) \left(\frac{v_0 + V\theta - s}{R(S)} \right)^\alpha f(v_0 + V\theta)\, \mathrm{d}\theta \\ &= \exp(\mathrm{i}t_\perp^T v_0) G_\alpha(d, \Delta, f, t_\parallel). \end{aligned} \tag{28}$$

Here $t_\parallel = V(V^T V)^{-1} V^T t$ is parallel to H and $t = t_\perp + t_\parallel$. The perpendicular component $t_\perp = t - t_\parallel$ is in the nullspace of V^T and hence factors through the integral over Δ_0. We compute the parallel moments $G(d, \Delta, f, t_\parallel)$ by numerical quadrature if the parallel variation is small and recurrence otherwise.

4.1.2. Quadratures for small parallel variation

When $\|V^T t_\parallel\| = \|V^T t\|$ is small, the exponential factor $\exp(\mathrm{i}t_\parallel^T \sigma)$ is accurately approximated on Δ by a low-degree polynomial. If $\|V^T t\| \leq \epsilon$ and $\sigma = v_0 + V\theta \in \Delta$ then Taylor expansion gives

$$\exp(\mathrm{i}t_\parallel^T \sigma) = \exp(\mathrm{i}t_\parallel^T v_0) \exp(\mathrm{i}t_\parallel^T (\sigma - v_0)) = \exp(\mathrm{i}t_\parallel^T v_0)(1 + \mathrm{i}(V^T t)^T \theta) + O(\epsilon^2).$$

Suppose a quadrature rule

$$\int_\Delta g(\sigma)\, \mathrm{d}\sigma = \sum_{j=1}^{Q} w_j g(\sigma_j) + E_Q(g)$$

is exact ($E_Q(g) = 0$) for the $Q = \binom{p+m+q+d}{d}$ monomials g of degree $p + m + q$. Then it yields G with $O(\|V^T t\|^{q+1})$ accuracy in $O(QM)$ work. Such rules have been extensively developed [28]. We employ equidistant Grundmann-Moeller rules [29], which can easily be generated in arbitrary simplex dimension and degree of exactness. Many other simplex rules, with desirable properties such as positive weights and higher order, are available for special cases [30, 31, 32].

Specification of equidistant rules also suggests equidistant Lagrange representation, which parametrizes degree-p polynomials f on a d-dimensional simplex Δ by $P = \binom{p+d}{d}$ values $f(\sigma_\alpha)$ at equidistant points $\sigma_\alpha \in \Delta$ (Fig. 8). Equidistant Lagrange representation simplifies common linear operations such as

- evaluation at the Q equidistant points of Grundmann-Moeller rules,
- transformation to the standard simplex Δ_0,
- differentiation,
- restriction to simplex boundaries,
- multiplication by shifted monomials $(\sigma - s)^\alpha$.

Representation by values also permits the approximate evaluation of Fourier transform (1) when the densities f_i are arbitrary continuous functions, rather than polynomials.[3]

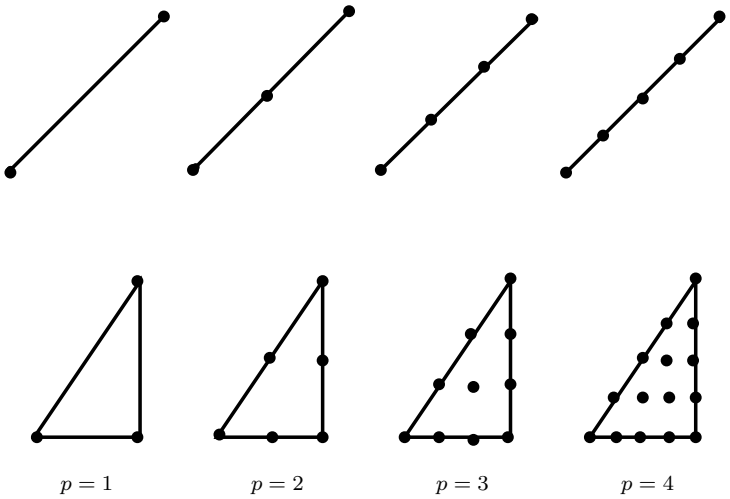


Figure 8: $P = \binom{p+d}{d} = 2, 3, 4, 5, 3, 6, 10, 15$ equispaced points for the representation of polynomials of degree $p = 1$ through 4 on simplices of dimension $d = 1$ and 2.

[3]The degree p of our polynomials is typically less than 6, reducing inaccuracy due to the Runge phenomenon [33]. It would be straightforward to employ better-conditioned evaluation points if desired.

4.1.3. Dimensional recurrences for large parallel variation

If $\|V^T t\| > \epsilon$, then we compute the parallel moments

$$G_\alpha(d, \Delta, f, t_\parallel) = \int_\Delta \exp(\mathrm{i} t_\parallel^T \sigma) \left(\frac{\sigma - s}{R(S)}\right)^\alpha f(\sigma)\, \mathrm{d}\sigma \tag{29}$$

via dimensional recurrences, derived from the Gauss formula for multidimensional integration by parts. Given a vector h parallel to the d-dimensional affine hyperplane H containing Δ, and a smooth function φ on $\mathbf{R}^D$, the Gauss formula reads

$$\int_\Delta h^T \nabla\varphi(\sigma)\, \mathrm{d}\sigma = \int_{\partial\Delta} h^T n\, \varphi(\sigma)\, \mathrm{d}\sigma.$$

The boundary $\partial\Delta = \cup_{j=0}^d \partial_j\Delta$ and outward unit normal n of Δ are defined relative to H (Fig. 9). The Gauss formula leads to efficient recurrences (f and g), which reduce moments in dimension d to dimension $d-1$ and terminate at $d = 0$.

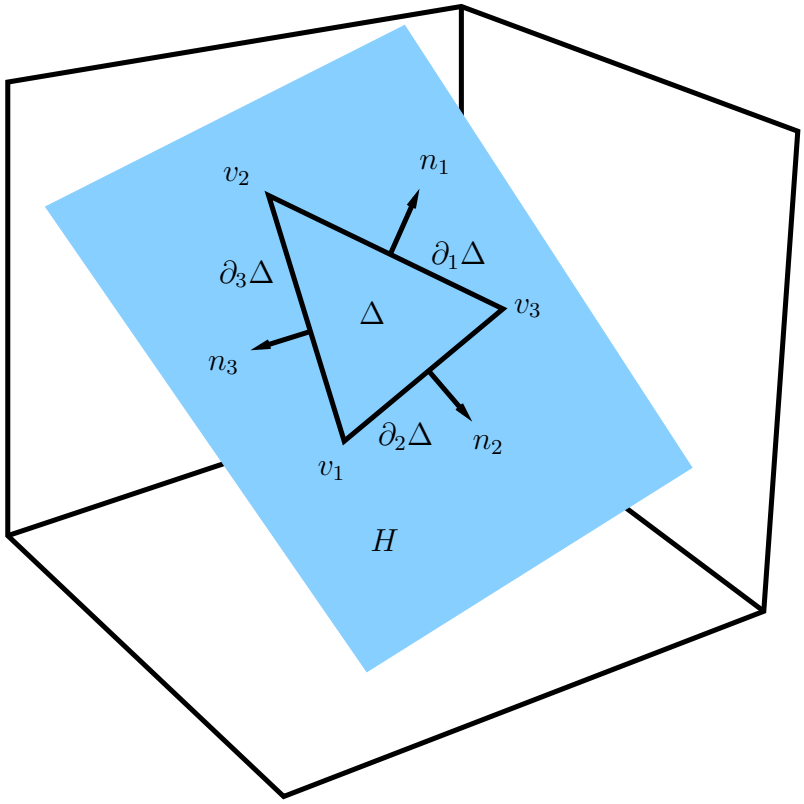


Figure 9: Vertices v_j and outward unit normals n_j relative to the affine hyperplane H, for a simplex Δ with $d = 2$ in ambient dimension $D = 3$.

The f recurrence. For the triple product $e(\sigma)f(\sigma)g(\sigma) = \exp(\mathrm{i}t^T\sigma)f(\sigma)(\sigma - s)^\alpha$, where $\nabla e(\sigma) = \mathrm{i}te(\sigma)$ and $g(\sigma) = ((\sigma - s)/R(S))^\alpha$, the Gauss formula gives

$$\begin{aligned}\int_\Delta h^T\nabla(efg) &= \int_{\partial\Delta} h^T nefg \\ &= \int_\Delta \mathrm{i}h^T tefg + eh^T\nabla fg + efh^T\nabla g. \end{aligned} \tag{30}$$

Solving Eq. (30) for $\int_\Delta efg$ gives

$$\int_\Delta efg = \frac{1}{\mathrm{i}h^T t}\left[\int_{\partial\Delta} h^T nefg - \int_\Delta eh^T\nabla fg - \int_\Delta efh^T\nabla g\right],$$

and setting $z = -\mathrm{i}h/h^T t$ gives

$$\int_\Delta e((I + z^T\nabla)f)g = \int_{\partial\Delta} z^T n\ efg - \int_\Delta efz^T\nabla g. \tag{31}$$

Since Eq. (31) holds for all polynomials f and g, it also holds with f replaced by the degree-p polynomial[4]

$$f_1 = \left(I + z^T\nabla\right)^{-1} f = \left(I - z^T\nabla + (z^T\nabla)^2 - \cdots + (-1)^p(z^T\nabla)^p\right) f,$$

which satisfies $(I + z^T\nabla)f_1 = f$. Thus Eq. (31) yields

$$\int_\Delta efg = \int_{\partial\Delta} z^T n\ ef_1 g - \int_\Delta ef_1 z^T\nabla g = \int_{\partial\Delta} z^T n\ ef_1 g_0 + \int_\Delta ef_1 g_1 \tag{32}$$

where $g_0 = g$ and $g_r = (-z^T\nabla)^r g$ is a polynomial of degree $\le m - r$ for $r \ge 1$. Since $g_{m+1} = 0$, iterating Eq. (32) eliminates the integrals over Δ and yields

$$\int_\Delta efg = \int_{\partial\Delta} z^T n\ e\left(f_1 g_0 + f_2 g_1 + \cdots + f_{m+1} g_m\right) \tag{33}$$

where $f_r = (I + z^T\nabla)^{-r} f$ for $r \ge 1$. Each g_r moment is a linear combination of moments of f_{r+1} with order $m - r$. Thus Eq. (33) expresses order-m moments of a degree-p polynomial f over a d-dimensional simplex Δ, as linear combinations of order-$(m - r)$ moments of $m+1$ degree-p polynomials f_{r+1} over $d+1$ lower-dimensional simplices $\partial_k\Delta$. Here the boundary $\partial\Delta = \cup_{k=0}^d \partial_k\Delta$ of Δ in H consists of $d + 1$ oriented simplices of dimension $d - 1$ given by (Fig. 9)

$$\partial_k\Delta = \left\{\sum_{j\ne k} \theta_j v_j \mid \theta_j \ge 0, \sum_{j\ne k} \theta_j = 1\right\}. \tag{34}$$

[4]On the P-dimensional space of polynomials of degree $\le p$, the operator $z^T\nabla$ satisfies $(z^T\nabla)^{p+1} = 0$. Hence all eigenvalues of $z^T\nabla$ are 0, $I + z^T\nabla$ is invertible, and the geometric series terminates.

To make Eq. (33) more explicit, define the D-vector of moment shift operators $E = (E_1, \ldots, E_D)$ on $\mathbf{C}^M$ satisfying

$$z^T \nabla \left(\frac{\sigma - s}{R(S)}\right)^\alpha = z^T E \left(\frac{\sigma - s}{R(S)}\right)^\alpha = \frac{1}{R(S)} \sum_{j=1}^{D} z_j \alpha_j \left(\frac{\sigma - s}{R(S)}\right)^{\alpha - e_j}$$

and

$$z^T E G_\alpha(d, \Delta, f, t) = \frac{1}{R(S)} \sum_{j=1}^{D} z_j \alpha_j G_{\alpha - e_j}(d, \Delta, f, t).$$

Then Eq. (33) becomes the f recurrence

$$\begin{aligned} G(d, \Delta, f, t) &= \sum_{k=0}^{d} z^T n_k (G(d-1, \partial_k \Delta, f_1, t) - z^T E G(d-1, \partial_k \Delta, f_2, t) \\ &+ (-z^T E)^2 G(d-1, \partial_k \Delta, f_3, t) + \cdots \\ &+ (-z^T E)^m G(d-1, \partial_k \Delta, f_{m+1}, t)) \\ &= \sum_{k=0}^{d} z^T n_k \sum_{r=0}^{m} (-z^T E)^r G(d-1, \partial_k \Delta, f_{r+1}, t). \end{aligned} \tag{35}$$

The g recurrence. Interchanging the roles of f and g in Eq. (31) leads to a shorter but more complicated recurrence

$$\int_\Delta efg = \int_{\partial\Delta} z^T n \; e \left(g^1 f^0 + g^2 f^1 + \cdots + g^{p+1} f^p\right) \tag{36}$$

if $p < m$. Here

$$g^r = (I + z^T \nabla)^{-r} g, \qquad f^r = (-z^T \nabla)^r f.$$

Each g^{r+1} moment is a linear combination of moments of f^r with order m. Thus Eq. (36) expresses order-m moments of a degree-p polynomial f over a d-dimensional simplex Δ, as linear combinations of order-m moments of $p+1$ degree-$(p-r)$ polynomials f^r over the $d+1$ lower-dimensional simplices $\partial_k \Delta$ of Eq. (34). Eq. (36) becomes the g recurrence

$$\begin{aligned} G(d, \Delta, f, t) &= \sum_{k=0}^{d} z^T n_k ((I + z^T E)^{-1} G(d-1, \partial_k \Delta, f^0, t) \\ &+ (I + z^T E)^{-2} G(d-1, \partial_k \Delta, f^1, t) + \cdots \\ &+ (I + z^T E)^{-p-1} G(d-1, \partial_k \Delta, f^p, t)) \\ &= \sum_{k=0}^{d} z^T n_k \sum_{r=0}^{p} (I + z^T E)^{-r-1} G(d-1, \partial_k \Delta, f^r, t). \end{aligned} \tag{37}$$

Here $I + z^T E$ is invertible since all eigenvalues of E are 0. Algorithm 3 computes G by quadrature and recurrence.

Algorithm 3 Computation of $G(d, \Delta, f, t)$ by quadrature and recurrence.

if $d = 0$

  $G(d, \Delta, f, t) = \left(\frac{\sigma - s}{R(S)}\right)^{\alpha} \exp(\mathrm{i}t^T\sigma)f$

else if $\|V^T t\| \leq \epsilon$

  Generate quadrature rule of order $p + m + q$

  Approximate $G(d, \Delta, f, t)$ to order $O(\epsilon^{q+1})$ by quadrature

$t_\perp = 0$

if $0 < d < D$

  Extract parallel variation $t_\parallel = V(V^T V)^{-1} V^T t$

  $t = t_\parallel$

if $p < m$

  $z = -\mathrm{i}t/\|t\|^2$

  $f^0 = f$

  $G = 0$

  for $r = 0 \ldots p$

    for $k = 0 \ldots d$

      $G = G + z^T n_k (I + z^T E)^{-r-1} G(d - 1, \partial_k \Delta, f^r, t)$

    $f^{r+1} = (-z^T \nabla) f^r$

else

  $z = -\mathrm{i}t/\|t\|^2$

  $f_0 = f$

  $G = 0$

  for $r = 0 \ldots m$

    $f_{r+1} = (I + z^T \nabla) f_r$

    for $k = 0 \ldots d$

      $G = G + z^T n_k (-z^T E)^r G(d - 1, \partial_k \Delta, f_{r+1}, t)$

$G(d, \Delta, f, t) = \exp(\mathrm{i}t_\perp^T v_0) G$

Direct evaluation. Our dimensional recurrences provide an $O(N^2)$ direct algorithm for evaluating the piecewise-polynomial Fourier transform (1):

$$\widehat{f}(\tau_k) = \sum_{i=1}^{N} \int_{\Delta_i} \exp(\mathrm{i}\tau_k^T \sigma) f_i(\sigma)\, \mathrm{d}\sigma = \sum_{i=1}^{N} G_0(d, \Delta_i, f_i, \tau_k).$$

The f recurrence (35) expresses each $G_0(d, \Delta, f, t)$ in terms of lower-dimensional moments of another polynomial $f_1 = \left(I + z^T \nabla\right)^{-1} f$:

$$G_0(d, \Delta, f, t) = \sum_{k=0}^{d} z^T n_k \; G_0(d-1, \partial_k \Delta, f_1, t). \tag{38}$$

The recurrence (38) terminates when $d = 0$, as Δ becomes a point. Consequently, direct evaluation of the piecewise-polynomial Fourier transform (1) costs $(d+1)!$ pointwise Fourier transforms (3).

4.1.4. Cost and stability of Algorithm 3

Let $W(m, p, d, D)$ be the cost of computing M moments $G(d, \Delta, f, t)$ of a degree-p polynomial f on a d-dimensional simplex $\Delta \subset \mathbf{R}^D$. If $p < m$, the g recurrence (37) reduces d to $d-1$ in p $P \times P$ and $M \times M$ matrix-vector products. If $p \geq m$, the f recurrence (35) reduces d to $d-1$ in m $P \times P$ and $M \times M$ matrix-vector products. Accordingly, the total cost $W(m, p, d, D)$ is bounded by

$$\begin{aligned} W(m, p, d, D) &\leq (d+1) \min(m, p)(P^2 + M^2 + W(m, p, d-1, D)) \\ &\leq O((d+1)! \min(m, p)^d (P^2 + M^2)) \\ &\leq O(\min(m, p)^d (p^{2d} + m^{2D})). \end{aligned}$$

This bound overestimates the cost of computing G by Algorithm 3, since many branches terminate with quadrature.[5]

When $z = -\mathrm{i}h/h^T t_\parallel$ is small, the terms in the formally infinite sum defining $(I + z^T \nabla)^{-1}$ rapidly decrease to zero, stabilizing both f and g recurrences. The choice $h = t_\parallel$ makes $z = -\mathrm{i}t_\parallel / \|t_\parallel\|^2$ small when $t_\parallel$ is large. When $t_\parallel$ is small, the recurrences can be unstable but numerical quadrature takes up the slack (Section 4.1.2).

4.2. Hierarchical tree structures

Geometric objects such as simplices can be localized into hierarchical tree structures (Section 2.6) by approximation and remaindering (Fig. 10).

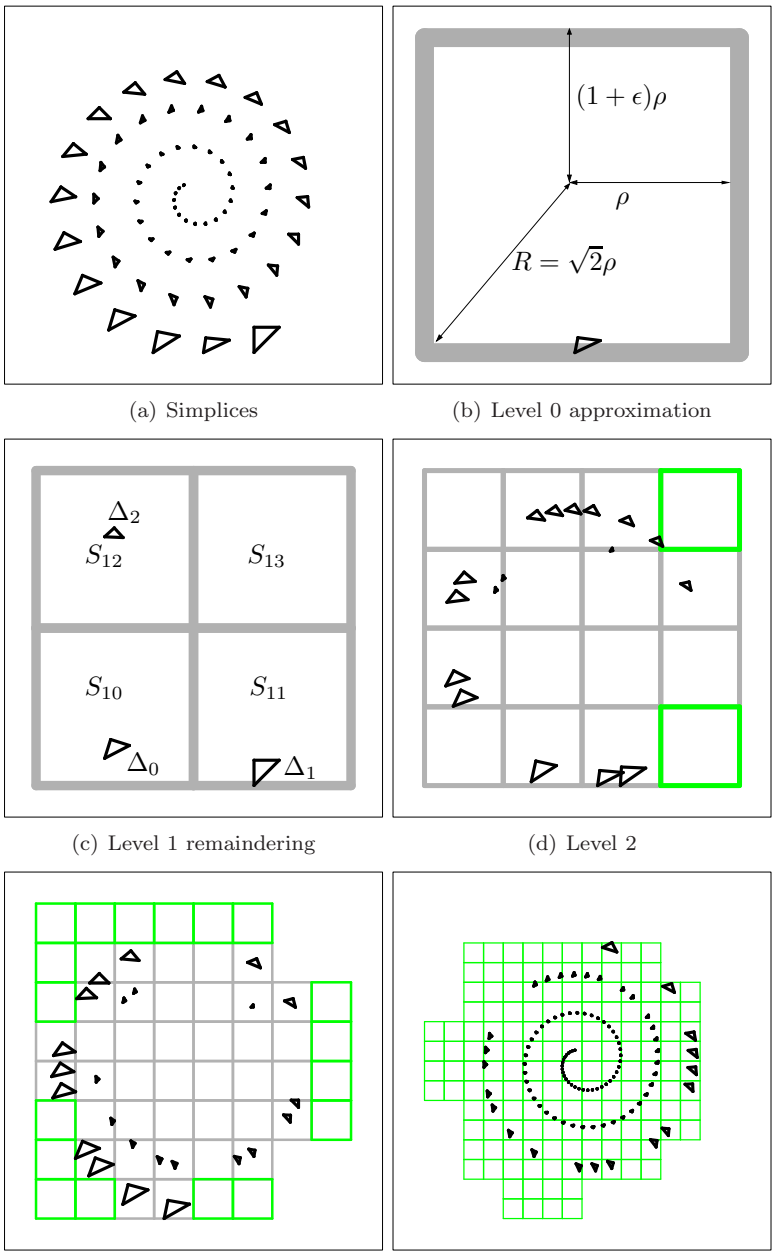
Approximation sorts simplices which overlap the boundaries of a cubical cell S_{lI}. We define a simplex Δ to fit within ϵ of a cell $s + \rho Q$ if each vertex v of Δ satisfies $|v_j - s_j| \leq (1 + \epsilon)\rho$ for $j = 1$ to D. All simplices are initially assigned to the root cell S_{00}. Then as the tree is constructed, simplices are sorted to fit within ϵ of child cells whenever possible.

[5]Since E and $z^T \nabla$ are essentially tensor products, M^2 can be replaced by $mM \log M = O(m^{D+1} \log m) = O(\log^{D+1} \epsilon)$.

Remaindering treats simplices which do not fit within ϵ of any leaf cell. The piecewise-polynomial algorithm assigns such simplices to the smallest possible nonleaf source child cell, and folds them into the coefficients at each level. After each split-merge step, source simplices Δ_i remaining in each source parent cell $S_{l-1,I}$ contribute to the coefficient vector for each target child cell $T_{L-l+1,K}$ via

$$C_\alpha(S_{l-1,I}, T_{L-l+1,K}) = C_\alpha(S_{l-1,I}, T_{L-l+1,K}) + \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \sum_{\Delta_i \subset S_{l-1,I}} G_\alpha(d, \Delta_i, f_i, t_{L-l+1,K}).$$

The cost of the algorithm is unaffected by remaindering if the number of remaindered simplices is $O(1)$. (Remaindered simplices can also be subdivided by the algorithm of [21]).



(a) Simplices (b) Level 0 approximation

(c) Level 1 remaindering (d) Level 2

(e) Level 3 (f) Level 4

Figure 10: Hierarchical tree structure with $d = D = 2$, $L = 4$ levels, approximation and remaindering. On level 0, a simplex fits within $\epsilon = 0.05$ of the source root cell. On level 1, simplex Δ_i is remaindered into source cell S_{1i} and Δ_1 fits within ϵ into S_{11}.

4.3. The piecewise-polynomial butterfly algorithm

Algorithm 4 evaluates the Fourier transform of N_S polynomials of degree p on N_S simplices in radius $R(S)$, at N_T targets τ_k in radius $R(T)$). The number of tree levels L satisfies $2^{-L}R(S)R(T) \leq R$ where $(Re/m)^m \leq \epsilon$. The algorithm proceeds in four steps with a total cost of $O(L2^{DL}M^2 + N_S M^2 + N_T M)$:

1. L-level hierarchical tree structures containing source simplices (within ϵ) and target points are constructed. Cost $O(2^{DL} + LN_S + LN_T)$ by Algorithm 1.
2. Dimensional recurrence and quadrature compute $M2^{DL}$ source-local coefficients, which represent the source simplices in 2^{DL} source leaf cells to the target root cell. Cost $O(N_S M^2)$ by Algorithm 3.
3. A butterfly scheme with $L2^{DL}$ split-merge steps transforms source-local to target-local coefficients, which represent all the source simplices to 2^{DL} target leaf cells. Remaindered simplices contribute at each step. Cost $O(L2^{DL}M^2)$.
4. An M-term expansion is evaluated at each of N_T targets. Cost $O(N_T M)$.

Algorithm 4 A piecewise-polynomial butterfly algorithm

Step 1 - Localization

Fit source simplices Δ_i and targets τ_k within ϵ into leaf cells of L-level trees $\mathcal{S}_L$ and $\mathcal{T}_L$

Step 2 - Compute source-local coefficients by Algorithm 3

for $I = 0 \dots 2^{DL} - 1$
  $S = S_{LI}$
  $s = s_{LI}$
  $T = T_{00}$
  $t = t_{00}$
  for $|\alpha| \le m$
    $C_\alpha(S,T) = \frac{\mathrm{i}^{|\alpha|}}{\alpha!} \sum_{\Delta_i \subset S} \int_{\Delta_i} \left(\frac{\sigma - s}{R(S)}\right)^\alpha \exp(\mathrm{i}t^T\sigma) f_i(\sigma)\,\mathrm{d}\sigma$

Step 3 - Butterfly scheme with remainder

for $l = L \dots 1$
  for $I = 0 \dots 2^{D(l-1)} - 1$
    $S = S_{l-1,I}$
    $s = s_{l-1,I}$
    $J = 2^D I$
    for $K = 0 \dots 2^{D(L-l)} - 1$
      $T = T_{L-l,K}$
      $t = t_{L-l,K}$
      for $k = 0 \dots 2^D - 1$
        $T_1 = T_{L-l+1,2^D K+k}$
        $t_1 = t_{L-l+1,2^D K+k}$
        $C(S,T_1) = \sum_{i=0}^{2^D-1} A(s_{l,J+i} - s) B(s_{l,J+i}, t_1 - t) C(S_{l,J+i}, T)$
        for $\Delta_i \subset S$
          for $|\alpha| \le m$
            $C_\alpha(S,T_1) = C_\alpha(S,T_1) + \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \int_{\Delta_i} \left(\frac{\sigma - s}{R(S)}\right)^\alpha \exp(\mathrm{i}t_1^T\sigma) f_i(\sigma)\,\mathrm{d}\sigma$

Step 4 - Evaluate target-local expansions

for $K = 0 \dots 2^{DL} - 1$
  $S = S_{00}$
  $s = s_{00}$
  $T = T_{LK}$
  $t = t_{LK}$
  for $\tau_k \in T$
    $\widehat{f}(\tau_k) = \exp(\mathrm{i}(\tau_k - t)^T s)) \sum_{|\alpha| \le m} C_\alpha(S,T) \left(\frac{\tau_k - t}{R(T)}\right)^\alpha$

5. Numerical results

The piecewise-polynomial butterfly algorithm (Algorithm 4) has been implemented in Fortran 77, for simplices of arbitrary dimension d in arbitrary ambient dimension D. Multidimensional arrays are mapped to one-dimensional arrays. Linear operators such as differentiation, integration and interpolation of polynomials are applied by matrix-vector multiplication with precomputed matrices. All numerical results were obtained with the `gfortran` compiler on a single Intel Xeon E5-2670 v2 processor with 4 GB of memory. The accuracy and efficiency of the implementation have been verified on a gallery of test cases including simplices of dimensions $d = 0$ (points) through $d = 2$ (triangles) in ambient dimensions $D = 1$ through $D = 3$. We report CPU times

- T_s for Algorithm 4 with $s = 3, 6, 9$ and 12-digit accuracy,
- T_d for direct evaluation (Section 4.1.3),
- T_F for a standard N_F-point complex D-dimensional FFT [34]. (The FFT timings T_F are monotonized by choosing N_F to be the smallest product of powers of 2, 3 and 5 which is larger than N.)

5.1. Discrete points in $\mathbf{R}^D$

Algorithm 4 evaluates the pointwise Fourier transform (3) when the source simplex dimension $d = 0$. We tested it on N random source and target points in radii $R(S) = O(1)$ and $R(T) = O(N^{1/D})$. The expected $N \log N$ scaling is clearly demonstrated by CPU times plotted in Fig. 11. In dimensions $D = 1$ (Table 3) and $D = 2$ (Table 4), Algorithm 4 is much faster than direct evaluation, for all problem sizes N and maximum twelve-digit accuracy. In dimension $D = 3$ (Table 5), it is faster for all problem sizes N at six-digit accuracy, while nine-digit accuracy breaks even at $N = 1728$.

Compared to the classical FFT for N_F equidistant points, the cost of Algorithm 4 is asymptotically within two orders of magnitude for six-digit accuracy. Doubling the accuracy multiplies the cost by roughly 2^D in D dimensions.

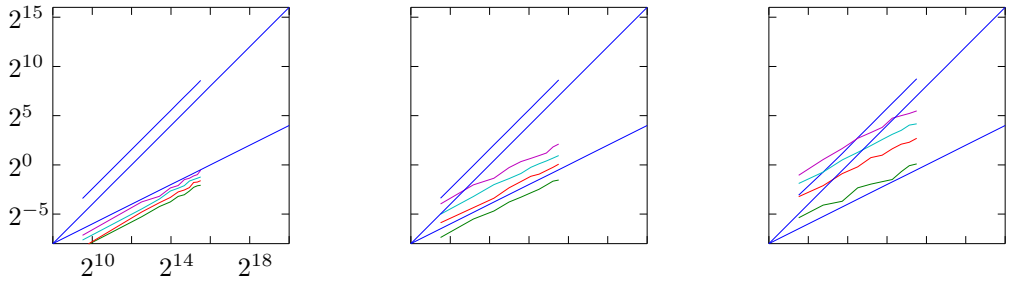


Figure 11: Direct and butterfly CPU time required to obtain 3, 6, 9 and 12-digit accuracy, vs. $N_T = N_S$ targets and $729 \leq N_S \leq 46656$ source points with $d = 0$ in ambient dimension (left to right) $1 \leq D \leq 3$.

Table 3: CPU times with $d = 0$ and $D = 1$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
729	0.00006	0.094	0.003	0.003	0.005	0.007
5832	0.00050	6.001	0.026	0.036	0.045	0.074
10935	0.00105	20.933	0.053	0.070	0.091	0.110
16038	0.00148	44.571	0.074	0.101	0.159	0.196
21141	0.00195	77.911	0.109	0.149	0.194	0.234
26244	0.00252	120.007	0.122	0.170	0.232	0.352
31347	0.00299	171.008	0.159	0.203	0.320	0.391
36450	0.00350	231.127	0.206	0.289	0.356	0.460
41553	0.00386	299.677	0.230	0.297	0.390	0.503
46656	0.00452	380.333	0.237	0.329	0.428	0.691

Table 4: CPU times with $d = 0$ and $D = 2$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
729	0.00011	0.096	0.006	0.017	0.031	0.064
2304	0.00038	0.951	0.022	0.049	0.108	0.244
4761	0.00078	4.107	0.039	0.096	0.249	0.392
8100	0.00152	11.773	0.073	0.198	0.379	0.827
12321	0.00234	27.512	0.103	0.308	0.546	1.260
17424	0.00322	54.663	0.140	0.447	0.843	1.577
23409	0.00437	99.085	0.179	0.520	1.079	1.921
30276	0.00596	165.424	0.241	0.669	1.316	2.300
38025	0.00728	260.633	0.318	0.843	1.603	3.505
46656	0.00898	394.457	0.343	1.051	1.920	4.339

Table 5: CPU times with $d = 0$ and $D = 3$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
729	0.00017	0.119	0.024	0.107	0.271	0.480
1728	0.00044	0.600	0.059	0.232	0.604	1.455
3375	0.00087	2.228	0.077	0.541	1.421	3.165
5832	0.00156	6.742	0.199	0.869	2.414	6.492
9261	0.00271	16.878	0.258	1.668	3.903	9.729
13824	0.00388	37.682	0.305	1.977	5.956	13.989
19683	0.00586	77.079	0.356	3.094	8.576	26.750
27000	0.00806	143.503	0.616	4.295	11.370	31.995
35937	0.00996	254.963	0.941	4.897	16.429	37.301
46656	0.01377	429.199	1.073	6.510	17.947	44.838

5.2. Line segments in $\mathbf{R}^D$

We tested the piecewise-polynomial butterfly algorithm (Algorithm 4) on N_S randomly placed line segments in $\mathbf{R}^D$, setting each f_i to a random cubic polynomial and evaluating the piecewise-polynomial Fourier transform $\widehat{f}(\tau_k)$ given by Eq. (4) at N random targets $\tau_k \in \mathbf{R}^D$. Here $N = 4N_S$ is the total number of degrees of freedom in the input. Fig. 12 clearly demonstrates $N \log N$ scaling.

The algorithm runs much faster than direct evaluation (Tables 6–8). For twelve-digit accuracy, it outpaces direct evaluation for every problem size N, by several orders of magnitude for large N. Since the number of degrees of freedom is larger, the algorithm compares favorably with the standard uniform FFT: For twelve-digit accuracy in $D \leq 2$, or six-digit accuracy in $D = 3$, it runs as fast as 100 FFTs. As in the pointwise case, doubling the accuracy multiplies the cost by roughly 2^D in D dimensions.

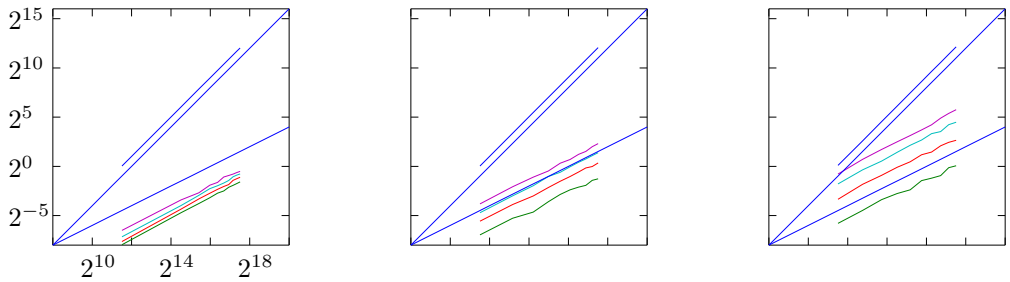


Figure 12: Direct and butterfly CPU time required to obtain 3, 6, 9 and 12-digit accuracy, vs. $N_T = 4N_S$ targets and $729 \leq N_S \leq 46656$ source segments with $d = 1$ in ambient dimension (left to right) $1 \leq D \leq 3$.

Table 6: CPU times with $d = 1$ and $D = 1$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
2916	0.00026	1.027	0.004	0.005	0.007	0.011
23328	0.00221	65.436	0.038	0.049	0.063	0.094
43740	0.00428	231.695	0.073	0.101	0.130	0.156
64152	0.00645	496.861	0.110	0.150	0.208	0.265
84564	0.00850	859.472	0.152	0.198	0.255	0.323
104976	0.01064	1324.053	0.177	0.243	0.309	0.471
125388	0.01260	1888.779	0.230	0.276	0.369	0.525
145800	0.01416	2546.873	0.263	0.373	0.481	0.582
166212	0.01738	3326.878	0.298	0.418	0.539	0.646
186624	0.01777	4176.714	0.334	0.463	0.586	0.704

Table 7: CPU times with $d = 1$ and $D = 2$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
2916	0.00047	1.025	0.008	0.021	0.038	0.071
9216	0.00156	10.328	0.026	0.068	0.125	0.238
19044	0.00352	44.402	0.040	0.124	0.256	0.466
32400	0.00664	128.382	0.083	0.226	0.490	0.721
49284	0.01016	296.955	0.138	0.351	0.659	1.243
69696	0.01328	592.824	0.191	0.490	0.974	1.618
93636	0.01836	1072.379	0.232	0.669	1.324	2.315
121104	0.02344	1797.046	0.262	0.892	1.686	2.870
152100	0.03125	2834.051	0.372	0.988	2.123	3.979
186624	0.03516	4268.295	0.416	1.273	2.607	4.947

Table 8: CPU times with $d = 1$ and $D = 3$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
2916	0.00071	1.068	0.018	0.098	0.289	0.580
6912	0.00180	6.008	0.045	0.281	0.781	1.631
13500	0.00352	23.203	0.096	0.529	1.445	3.262
23328	0.00703	69.483	0.148	0.932	2.595	5.633
37044	0.01055	175.814	0.191	1.381	4.410	8.756
55296	0.01641	393.930	0.365	2.260	6.171	13.045
78732	0.02461	799.237	0.436	2.729	9.301	24.827
108000	0.03398	1508.203	0.520	4.270	11.570	29.903
143748	0.04336	2679.131	0.912	5.241	18.248	41.703
186624	0.05273	4530.735	1.047	6.006	21.025	49.415

5.3. Triangles in $\mathbf{R}^D$

We tested the piecewise-polynomial butterfly algorithm (Algorithm 4) on N_S randomly placed triangles in $\mathbf{R}^D$, setting each f_i to a random cubic polynomial and evaluating the piecewise-polynomial Fourier transform $\widehat{f}(\tau_k)$ given by Eq. (5) at N random targets $\tau_k \in \mathbf{R}^D$. Here $N = 10N_S$ is the total number of degrees of freedom in the input.

The algorithm runs much faster than direct evaluation (Tables 9 and 10). For six-digit accuracy, it outpaces direct evaluation by three orders of magnitude when $N \geq 47610$. It requires less CPU time than 10 FFTs to obtain three-digit accuracy in ambient dimensions $D = 2$ and $D = 3$. The CPU time clearly scales as $O(N \log N)$ (Fig. 13).

Table 9: CPU times with $d = 2$ and $D = 2$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
7290	0.00131	5.946	0.012	0.029	0.060	0.103
23040	0.00449	59.958	0.031	0.093	0.180	0.292
47610	0.00933	256.398	0.075	0.177	0.349	0.603
81000	0.01758	741.004	0.120	0.302	0.613	0.941
123210	0.02578	1712.487	0.145	0.457	0.958	1.530
174240	0.03398	3430.010	0.228	0.673	1.330	2.064
234090	0.04961	6202.242	0.270	0.794	1.604	2.696
302760	0.06641	10374.852	0.408	1.113	2.186	3.469
380250	0.08438	16010.604	0.512	1.312	2.896	4.312
466560	0.10000	24149.947	0.562	1.766	3.125	5.090

Table 10: CPU times with $d = 2$ and $D = 3$.

N	T_F	T_d	T_3	T_6	T_9	T_{12}
7290	0.00180	5.923	0.023	0.117	0.391	0.652
17280	0.00469	34.358	0.059	0.312	0.812	1.930
33750	0.00937	132.627	0.090	0.672	1.719	3.762
58320	0.01641	400.494	0.180	0.922	3.059	6.414
92610	0.02813	994.472	0.230	1.297	5.059	9.566
138240	0.04219	2231.280	0.297	2.098	6.824	15.309
196830	0.06094	4623.198	0.598	3.051	8.539	23.145
270000	0.08965	8701.831	1.044	3.691	13.785	44.735
359370	0.11660	15547.841	1.237	4.405	15.828	55.374
466560	0.15469	26300.953	1.489	9.431	27.575	66.321

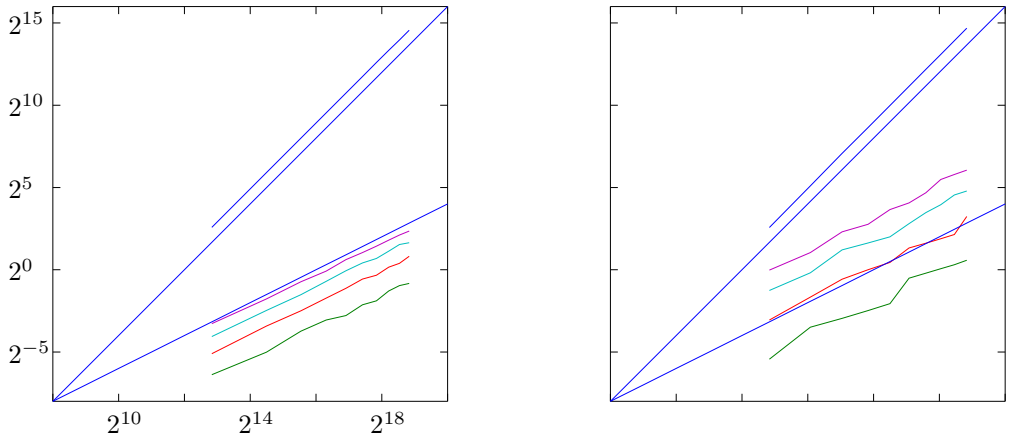


Figure 13: Direct and butterfly CPU time required to obtain 3, 6, 9 and 12-digit accuracy, vs. $N_T = 10N_S$ targets and $729 \leq N_S \leq 46656$ source triangles with $d = 2$ in ambient dimension (left to right) $2 \leq D \leq 3$.

6. Extensions and applications

The piecewise-polynomial butterfly algorithm (Algorithm 4) could be accelerated by changing expansion basis functions, extended to the application of Galerkin matrices, and generalized to evaluate Laplace and Gauss transforms.

6.1. Changing basis functions

Butterfly algorithms are often accelerated by changing basis functions. For example, replacing Taylor by Chebyshev expansions gives equal accuracy with fewer terms, leading to a factor of 2 speedup in the one-dimensional pointwise butterfly algorithm of [4].

Implicit change of basis functions can be implemented via a linear-algebraic viewpoint [35]: factorize the matrix $U_{ki} = \exp \mathrm{i}\tau_k^T \sigma_i$ into $U = EW_L W_{L-1} \cdots W_1 F$. Here F sums basis functions over sources to compute coefficients and E evaluates expansions at targets. Sparse block matrices W_p split and merge the coefficients. Rank-revealing factorizations of W_p, such as the singular value or interpolative decompositions, can save computational effort. Interpolative decompositions factorize $W_p = X_p Y_p Z_p$, with X_p and Z_p submatrices of W_p, and Y_p small and well-conditioned. They implement a change of basis

$$U = (EX_L Y_L) Z_L X_{L-1} Y_{L-1} \cdots Z_2 X_1 (Y_1 Z_1 F),$$

which implicitly computes coefficients with $Y_1 Z_1 F$, splits and merges with small matrices $Z_p X_{p-1}$ and Y_{p-1}, and evaluates expansions with $EX_L Y_L$.

6.2. Galerkin matrices

Algorithm 4 extends to the fast application of Galerkin matrices [36]

$$\widehat{f}_k = \int_{\Lambda_k} g_k(\tau) \sum_{i=1}^{N} \int_{\Delta_i} \exp(\mathrm{i}\tau^T \sigma) f_i(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau, \tag{39}$$

where $\Lambda_k \subset \mathbf{R}^D$ are d-dimensional target simplices and g_k are degree-p polynomials. The low-rank kernel approximation (13) separates the variables τ and σ with controllable error F_m:

$$\begin{aligned} \widehat{f}_k \quad = \quad & \sum_{|\alpha| \le m} \int_{\Lambda_k} \left(\frac{\tau - t}{R(T)} \right)^{\alpha} \exp(\mathrm{i}(\tau - t)^T s) g_k(\tau) \, \mathrm{d}\tau \\ & \sum_{i=1}^{N} \frac{(\mathrm{i}R)^{|\alpha|}}{\alpha!} \exp(\mathrm{i}t^T s) \int_{\Delta_i} \left(\frac{\sigma - s}{R(S)} \right)^{\alpha} \exp(\mathrm{i}t^T (\sigma - s)) f_i(\sigma) \, \mathrm{d}\sigma + F_m. \end{aligned}$$

Summing over i produces source-local coefficients (Algorithm 4, Step 2). The butterfly scheme (Step 3) converts to target-local expansions. Evaluation (Step 4) is replaced by moment evaluation (Algorithm 3), with sources replaced by targets.

The algorithm just described requires source and target simplices to satisfy a slightly more restrictive version of inequality (12):

$$|(\tau - t)^T (\sigma - s)| \le R \quad \text{where} \quad \left(\frac{R\mathrm{e}}{m} \right)^m \le \epsilon$$

whenever S is a source cell with center s, T is a target cell with center t, $\tau \in \Lambda_k \subset T$, and $\sigma \in \Delta_j \subset S$. Large remaindered simplices above the middle level $L/2$ in both source and target space could violate this condition. Simplex subdivision as in [21] removes this restriction by making simplices smaller.

6.3. Laplace and Gauss transforms

Algorithm 4 extends to evaluate the Laplace transform

$$\tilde{f}(\tau_k) = \sum_{i=1}^{N} f_i \exp(-\tau_k^T \sigma_i).$$

The exponential kernel is approximated by multidimensional Taylor expansion

$$\exp(-t^T s) = \sum_{n=0}^{\infty} \frac{(-1)^n \left(\sum_{j=1}^{D} t_j s_j\right)^n}{n!} = \sum_{\alpha \geq 0} \frac{(-1)^{|\alpha|}}{\alpha!} t^\alpha s^\alpha.$$

The error bound (8) is invariant under rotation in the complex plane, so the error is bounded by $(Re/m)^m$ as long as $|t^T s| \leq R$. Accordingly, our butterfly algorithm also approximates the piecewise-polynomial Laplace transform

$$\widehat{f}_k = \int_{\Lambda_k} g_k(\tau) \sum_{i=1}^{N} \int_{\Delta_i} \exp(-\tau^T \sigma) f_i(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau \tag{40}$$

and generalizes the one-dimensional pointwise algorithms of [37, 38, 39].

The pointwise Gauss transform is given by

$$\widehat{f}(\tau_k) = \sum_{i=1}^{N} \exp(-\|\tau_k - \sigma_i\|^2) f_i, \qquad 1 \leq k \leq N,$$

where $\|\|$ is the Euclidean norm. Since

$$\exp(-\|\tau_k - \sigma_i\|^2) = \exp(-\|\tau_k\|^2) \exp(2\tau_k^T \sigma_i) \exp(-\|\sigma_i\|^2),$$

the Gauss transform is a pre- and post-processed Laplace transform. Thus our butterfly algorithm can approximate the piecewise-polynomial Gauss transform

$$\widehat{f}_k = \int_{\Lambda_k} g_k(\tau) \sum_{i=1}^{N} \int_{\Delta_i} \exp(-\|\tau - \sigma\|^2) f_i(\sigma) \, \mathrm{d}\sigma \, \mathrm{d}\tau \tag{41}$$

and generalize the pointwise algorithms of [40, 41, 42, 43, 44] to the multidimensional piecewise-polynomial setting. [6]

[6] The only previous piecewise-polynomial Gauss transform we are aware of is the 2D piecewise-cubic algorithm of [45].

7. Acknowledgments

This work was supported by the Air Force Office of Scientific Research grant number FA9550-11-1-0242.

8. References

[1] E. Michielssen, A. Boag, A multilevel matrix decomposition algorithm for analyzing scattering from large structures, IEEE Trans. Antennas Propagat. 44 (1996) 1086–1093.
[2] E. J. Candes, L. Demanet, L. Ying, A fast butterfly algorithm for the computation of fourier integral operators, SIAM J. Mult. Model. Simul. 7 (2008) 1727–1750.
[3] L. Ying, Sparse Fourier transform via butterfly algorithm, SIAM. J. Sci. Comput. 31 (2009) 1678–1694.
[4] M. O'Neil, F. Woolfe, V. Rokhlin, An algorithm for the rapid evaluation of special function transforms, Appl. Comput. Harmon. Anal. 28 (2010) 203–226.
[5] Y. Zhang, J. Liu, E. Kultursay, M. Kandemir, N. Pitsianis, X. Sun, Scalable parallelization strategies to accelerate NuFFT data translation on multicores, in: EuroPar 2010, Part II, LNCS 6272, Vol. 6272 of LNCS, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 125–136.
[6] L. Demanet, M. Ferrara, N. Maxwell, J. Poulson, L. Ying, A butterfly algorithm for synthetic aperture radar imaging, SIAM J. Imag. Sci. 5 (2012) 203–243.
[7] S. Kunis, I. Melzer, A stable and accurate butterfly sparse Fourier transform, SIAM J. Numer. Anal. 50 (2012) 1777–1800.
[8] J. Poulson, L. Demanet, N. Maxwell, L. Ying, A parallel butterfly algorithm, SIAM J. Sci. Comput. 36 (2013) C49–C65.
[9] S. Börm, C. Börst, J. M. Melenk, An analysis of a butterfly algorithm, Comput. Math. Applics. 74 (2017) 2125–2143.
[10] J. W. Cooley, J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19 (1965) 297–301.
[11] A. Dutt, V. Rokhlin, Fast Fourier transforms for nonequispaced data, SIAM J. Sci. Comput. 14 (6) (1993) 1368–1393.
[12] A. Dutt, V. Rokhlin, Fast Fourier transforms for nonequispaced data. II, Appl. Comput. Harmon. Anal. 2 (1) (1995) 85–100.
[13] G. Beylkin, On the fast Fourier transform of functions with singularities, Appl. Comput. Harm. Analysis 2 (1995) 363–381.
[14] D. Potts, G. Steidl, M. Tasche, Fast Fourier transforms for nonequispaced data: a tutorial, in: Modern sampling theory, Birkhauser, 2001, pp. 247–270.
[15] L. Greengard, J.-Y. Lee, Accelerating the nonuniform fast Fourier transform, SIAM Review 46 (2004) 443–454.
[16] S. Jiang, D. Wang, X.-P. Wang, An efficient boundary integral scheme for the MBO threshold dynamics method via the NUFFT, J. Sci. Comput. (2017) 1–17.
[17] J. Song, Y. Liu, S. Gewalt, G. Cofer, G. Johnson, Q. Liu, Least-square NUFFT methods applied to 2-D and 3-D radially encoded MR image reconstruction, IEEE Trans. Biomed. Eng. 56 (2008) 1134–1142.
[18] W. Sierpinski, Sur une courbe cantorienne dont tout point est un point de ramification, C.R. Acad. Sci. Paris 160 (1915) 302–305.
[19] E. Sorets, Fast Fourier transforms of piecewise constant functions, J. Comput. Phys. 116 (1995) 369–379.
[20] J. Sun, H. Li, Generalized Fourier transform on an arbitrary triangular domain, Adv. Comp. Math. 22 (2005) 223–248.
[21] I. Sammis, J. Strain, A geometric nonuniform fast Fourier transform, J. Comput. Phys. 228 (2009) 7086–7108.
[22] J. Strain, Locally-corrected spectral methods and overdetermined elliptic systems, J. Comput. Phys 224 (2007) 1243–1254.
[23] B. Engquist, L. Ying, Fast directional multilevel algorithms for oscillatory kernels, SIAM. J. Sci. Comput. 29 (2007) 1710–1737.
[24] H. Samet, The design and analysis of spatial data structures, Addison-Wesley, Reading, Massachusetts, 1990.

[25] G.-C. Rota, W. G. Strang, A note on the joint spectral radius, Indag. Math. 22 (1960) 379–381.
[26] R. M. Jungers, The Joint Spectral Radius: Theory and Applications, Springer-Verlag, 2009.
[27] A. Ben-Israel, The change of variables formula using matrix volume, SIAM J. Matrix Analysis 21 (1999) 300–312.
[28] P. J. Davis, P. Rabinowitz, Methods of Numerical Integration, 2nd Edition, Computer science and applied mathematics, Academic Press, 1984.
[29] A. Grundmann, M. Moeller, Invariant integration formulas for the N-Simplex by combinatorial methods, SIAM J. Num. Analysis 15 (1978) 282–290.
[30] R. Cools, An encyclopaedia of cubature formulas, Journal of Complexity 19 (3) (2003) 445 – 453.
[31] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, CAMWA 59 (2010) 663–676.
[32] D. M. Williams, L. Shunn, A. Jameson, Symmetric quadrature rules for simplexes based on sphere close packed lattice arrangements, J. Comput. Appl. Math. 266 (2014) 18–38.
[33] C. Runge, Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten, Zeit. Math. Phys. 46 (1901) 224 – 243.
[34] P. N. Swartztrauber, FFTPACK – a package of Fortran subprograms for the fast Fourier transform of periodic and other symmetric sequences, Fortran code, National Center for Atmospheric Research (1985).
[35] Y. Li, H. Yang, E. R. Martin, K. L. Ho, L. Ying, Butterfly factorization, SIAM J. Multiscale. Mod. Simul. 13 (2015) 713–742.
[36] H. Cai, Y. Xu, A fast Fourier-Galerkin method for solving singular boundary integral equations, SIAM J. Num. Analysis 46 (2008) 1965–1984.
[37] V. Rokhlin, A fast algorithm for the discrete Laplace transformation, J. Complexity 4 (1988) 12–32.
[38] J. Strain, A fast Laplace transform based on Laguerre functions, Math. Comp. 58 (1992) 275–284.
[39] S. Kunis, I. Melzer, Fast evaluation of real and complex exponential sums, Electron. Trans. Numer. Anal. 46 (2017) 23–35.
[40] L. Greengard, J. Strain, The fast Gauss transform, SIAM J. Sci. Stat. Comput 12 (1991) 79–94.
[41] J. Strain, The fast Gauss transform with variable scales, SIAM J. Sci. Stat. Comput 12 (1991) 1131–1139.
[42] L. Greengard, X. Sun, A new version of the fast Gauss transform, Documenta Mathematica Extra Volume ICM 1998 III (1998) 575–584.
[43] K. Kobayashi, A remark on the fast Gauss transform, Publ. RIMS Kyoto Univ. 39 (2003) 785–796.
[44] S. Kunis, D. Potts, G. Steidl, Fast Gauss transforms with complex parameters using NFFTs, J. Numer. Math. 14 (2006) 295–303.
[45] J. Strain, Fast adaptive methods for the free-space heat equation, SIAM J. Sci. Comput. 15 (1994) 185–206.