We present an algorithm for the evaluation of the Fourier transform of piecewise constant functions of two variables. The algorithm overcomes the accuracy problems associated with computing the Fourier transform of discontinuous functions; in fact, its time complexity is $O(\log^3(1/\epsilon)N^2 + \log^2(1/\epsilon)N^2 \log N)$, where ϵ is the accuracy and N is the size of the problem. The algorithm is based on the Lagrange interpolation formula and the Green's theorem, which are used to preprocess the data before applying the Fast Fourier transform. It admits natural generalizations to higher dimensions and to piecewise smooth functions.

Fast Fourier Transforms of
Piecewise Constant Functions

Eugene Sorets
Research Report YALEU/DCS/RR-986
September 1993

The author was supported by the National Science Foundation under Grant DMS901213595.
Approved for public release: distribution is unlimited.
Keywords: *Fourier Transform, Fast Algorithms.*

Contents

1 Introduction

The Fast Fourier Transform (see, for example, [2, 3, 8]) is a ubiquitous tool of numerical analysis, essential in signal processing, electrical engineering, VLSI circuit modeling, medical imaging, and innumerable other fields. It is a very effective tool when the function to be transformed is smooth; however, if the function has a jump discontinuity, then the accuracy of the FFT is significantly reduced. This loss of accuracy is the result of the simple fact that FFT, from the mathematical point of view, is a collection of integrals evaluated with trapezoidal rule. Therefore, the numerical error that results from the discontinuity is on the order of n^{-1} for a problem of size n. Such errors make even single precision calculations, especially in higher dimensions, prohibitively costly. The cost can be reduced with the help of the Richardson extrapolation, but double precision calculations are still virtually impossible.

We construct an algorithm that successfully attacks this problem; for example, on a 512×512 array our implementation takes about 50 times the CPU time of one 512×512 FFT for single precision results and about 160 times for double precision.

The algorithm uses Green's theorem to replace the area integrals, that naturally occur in computing the two-dimensional Fourier transform of piecewise constant functions, with line integrals, thereby reducing significantly the number of nodes required for integration; after that the weights are redistributed onto the uniform grid with the help of the Lagrange interpolation formula, bringing the data to the form suitable for the FFT. A detailed description of the algorithm can be found in Section 4, and performance results in Section 5.

In this paper, we present the analysis in two dimensions under the assumption that the functions to be transformed are piecewise constant and that the discontinuities occur along a collection of polygons. Neither of these is a serious limitation; the algorithm generalizes quite naturally to higher dimensions and to curved boundaries. In Section 6 we describe a generalization of our algorithm to piecewise smooth functions.

Acknowledgements: The author would like to thank Professors Ronald Coifman and Vladimir Rokhlin for several useful discussions. He would also like to thank Professor Eytan Barouch for suggesting the problem and supplying the VLSI masks for the experiments.

2 Mathematical Preliminaries

2.1 Lagrange Interpolation

The Lagrange interpolation formula (see, for example, [1, 4]) approximates a function $f : \mathbb{R}^1 \to \mathbb{C}^1$ by the expression:

$$f(x) = \sum_{m=1}^{p} f(x_m) \prod_{\substack{n=1 \\ n \neq m}}^{p} \frac{x - x_n}{x_m - x_n} + R_p(x), \tag{1}$$

where $x_1 < \ldots < x_p$ are points on the real line and $R_p(x)$ is the error term. For each $m = 1, \ldots, p$, we will denote by δ_m the polynomial defined by the formula:

$$\delta_m(x) = \prod_{\substack{n=1 \\ n \neq m}}^{p} \frac{x - x_n}{x_m - x_n}, \tag{2}$$

and observe that

$$\delta_m(x_n) = \begin{cases} 1 & \text{if } m = n, \\ 0 & \text{if } m \neq n. \end{cases}$$

In view of (2), formula (1) can be rewritten as

$$f(x) = \sum_{m=1}^{p} f(x_m)\delta_m(x) + R_p(x). \tag{3}$$

$R_p(x)$ in equations (1) and (3) is the error term and

$$R_p(x) = \frac{f^{(p)}(\xi)}{p!} \prod_{n=1}^{p} (x - x_n) \qquad \text{for some} \quad \xi \in (x_1, x_p).$$

In our case, the nodes $x_1, \ldots, x_p$ are going to be uniformly spaced with the sampling interval $h > 0$, and the point x will lie in the interval of length h centered on the center of the interpolation window $[x_1, x_p]$. Furthermore, the function f will be of the form $f(x) = \exp(2\pi i k x)$. Under these conditions,

$$\begin{aligned} |R_p(x)| \quad &\leq \quad \begin{cases} \dfrac{2}{\pi} \cdot \dfrac{\left[\Gamma\left(\frac{p}{2} + 1\right)\right]^2}{p!} \cdot h^p (2\pi k)^p & \text{for odd } p, \\ \dfrac{\left[\Gamma\left(\frac{p}{2} + 1\right)\right]^2}{p!} \cdot h^p (2\pi k)^p & \text{for even } p \end{cases} \\ &\leq \quad \frac{\left[\Gamma\left(\frac{p}{2} + 1\right)\right]^2}{p!} \cdot h^p (2\pi k)^p, \end{aligned} \tag{4}$$

where $\Gamma(z)$ is the Gamma function: $\Gamma(z + 1) = z!$.

We will also make use of the two-dimensional Lagrange interpolation formula:

$$f(x, y) = \sum_{n=1}^{p} \sum_{m=1}^{p} \delta_m(x) \cdot \delta_n(y) \cdot f(x_m, y_n) + R_p(x, y). \tag{5}$$

Here the points $\{(x_m, y_n)\}_{m,n=1}^{p}$ are the interpolation nodes, the functions $\delta_1, \ldots, \delta_p$ are the same as in (2), and the error term, $R_p(x, y)$, admits the bound

$$|R_p(x, y)| \leq \frac{\left[\Gamma\left(\frac{p}{2} + 1\right)\right]^2}{p!} \cdot \left[h_x^p (2\pi k)^p + h_y^p (2\pi l)^p\right] \tag{6}$$

when the function $f(x, y)$ has the form $\exp(2\pi i(kx + ly))$ and the nodes are uniformly spaced with the sampling interval h_x along the x-axis and h_y along the y-axis.

2.2 Green's theorem

We will encounter area integrals of the form

$$\int_D e^{-2\pi imx} e^{-2\pi iny}\, dydx,$$

where D is a bounded domain in $\mathbb{R}^2$. We will replace them with line integrals using the following version of Green's theorem in the plane:

$$\int_D \frac{\partial Q}{\partial x}\, dydx = \int_\Gamma Q\, dy. \tag{7}$$

Here $Q : \mathbb{R}^2 \to \mathbb{C}^1$ is a function in $C^2(\bar{D})$ and Γ is the boundary of D traversed counter-clockwise. Applying (7) to $Q(x,y) = e^{-2\pi imx} e^{-2\pi iny}$, we arrive at a formula that will be important to us later:

$$\int_D e^{-2\pi imx} e^{-2\pi iny}\, dydx = \begin{cases} \dfrac{1}{-2\pi im} \displaystyle\int_\Gamma e^{-2\pi imx} e^{-2\pi iny}\, dy & \text{when } m \neq 0, \\ \displaystyle\int_\Gamma x e^{-2\pi iny}\, dy & \text{when } m = 0. \end{cases} \tag{8}$$

2.3 Gauss-Legendre Quadratures

Gauss-Legendre integration scheme (see, for example, [4, 7]) provides an approximation to the integral of a function $f : [0,1] \to \mathbb{C}^1$:

$$\int_0^1 f(t)\, dt = \sum_{k=1}^{q} f(t_k)\omega_k + E_q(f), \tag{9}$$

where the points $t_1, \ldots, t_q$ are the Gauss-Legendre nodes, the numbers $\omega_1, \ldots, \omega_q$ are the Gauss-Legendre weights, and $E_q(f)$ is the error term. The Gauss-Legendre nodes and weights are chosen to make the approximation (9) exact whenever the function f is a polynomial of degree less than $2q$. Consequently, the Gauss-Legendre scheme is effective for functions that are well approximated by polynomials; in fact:

$$E_q(f) = \frac{2^{q+1}(q!)^4}{(2q+1)[(2q)!]^3} \cdot f^{(2q)}(\xi) \qquad \text{for some } \xi \in [0,1]. \tag{10}$$

The proof of (10) can be found in [4].

3 Mathematical Description of the Algorithm

3.1 Statement of the Problem

Our goal in this paper is to find a way to compute Fourier Transforms of piecewise constant functions in $\mathbb{R}^2$. In other words, we would like to construct an algorithm that for every

piecewise constant function $f : \mathbb{R}^2 \to \mathbb{C}^1$ will compute an approximation to the collection of integrals, to which we will refer as the Fourier Transform $\hat{f}$ of f:

$$\hat{f}(m,n) \stackrel{\text{def}}{=} \int_0^1 \int_0^1 f(x,y) e^{-2\pi i m x} e^{-2\pi i n y} \, dy dx, \tag{11}$$

where m and n are integers such that $-M < m \leq M$ and $-N < n \leq N$. We will refer to the pair (m,n) as the frequency. The pair (M,N), thus, is the highest frequency of interest. The most common approximation to the integrals (11), the Discrete Fourier Transform [2] is, essentially, the trapezoidal rule; its standard implementation is the Fast Fourier Transform (see, for example, [2, 3, 8]). For discontinuous functions the trapezoidal rule has accuracy on the order of N^{-1} for an N by N problem, or, conversely, for accuracy ϵ, will require on the order of ϵ^{-2} operations; such performance is woefully inadequate in most practical problems.

In this paper we solve the following problem:

Problem 1 *For every piecewise constant function f with discontinuities along a finite number of polygons, and for every accuracy $\epsilon > 0$, compute all the integrals (11) with accuracy ϵ in the number of operation not greater than*

$$O\left(\left(\log^3 \frac{1}{\epsilon}\right) \bar{N}^2 + \left(\log^2 \frac{1}{\epsilon}\right) \bar{N}^2 \log \bar{N}\right), \tag{12}$$

where $\bar{N} = \max\{M, N\}$.

Mathematical description and error analysis of the algorithm that solves Problem 1 can be found in §3.2 and §3.3, respectively, Section 4 contains the full description of the algorithm, and Section 5 contains the results of some of our numerical experiments. In Section 6 we describe an extension of the algorithm to piecewise smooth functions.

3.2 Description of the Algorithm

A piecewise-constant function $f : \mathbb{R}^2 \to \mathbb{C}^1$ can be written as a linear combination of characteristic functions. By the characteristic function $\mathbf{1}_S : \mathbb{R}^2 \to \mathbb{C}^1$ of a set $S \subset \mathbb{R}^2$ we mean, as usual:

$$\mathbf{1}_S(x,y) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } (x,y) \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Using this notation, we write the piecewise-constant f as:

$$f(x,y) = \sum_{j=1}^{J} K_j \cdot \mathbf{1}_{D_j}(x,y), \tag{13}$$

where J is a positive integer constant determined by the problem, and, for each j, K_j is a complex constant and D_j is a bounded domain in $\mathbb{R}^2$. We will assume from now on that all domains D_j lie within the unit square, their interiors are pairwise disjoint, and the boundaries Γ_j of the domains D_j are polygons. Indeed, for each j, let $\{\gamma_l^j : [0,1] \to \Gamma_j\}_{l=1}^{L_j}$ be the set of paths that form Γ_j. Then, our assumptions are:

1. $\bar{D}_j \subset [0,1] \times [0,1]$ for all j,

2. $\mathrm{Int}(D_j) \cap \mathrm{Int}(D_k) = \emptyset$ for all $j \neq k$.

3. Every path γ_l^j has the form:

$$\gamma_l^j(t) = (a_0 + at, b_0 + bt) \qquad \text{for } t \in [0,1]. \tag{14}$$

The first two assumptions do not restrict generality at all, and the third one is made to simplify the error estimates and the implementation. It does not affect the algorithm itself.

Substituting (13) into (11) and using (8), we get:

$$\begin{aligned} \hat{f}(m,n) &= \int_0^1 \int_0^1 \sum_{j=1}^{J} K_j \cdot \mathbf{1}_{D_j}(x,y) e^{-2\pi i m x} e^{-2\pi i n y}\, dy dx \\ &= \sum_{j=1}^{J} \int_{D_j} K_j \cdot e^{-2\pi i m x} e^{-2\pi i n y}\, dy dx \\ &= \sum_{j=1}^{J} \int_{\Gamma_j} K_j \cdot F_{m,n}(x,y)\, dy, \end{aligned} \tag{15}$$

where each Γ_j is the boundary of the domain D_j and the function $F_{m,n} : \mathbb{R}^2 \to \mathbb{C}^1$ is defined by the formula:

$$F_{m,n} \stackrel{\text{def}}{=} \begin{cases} \dfrac{e^{-2\pi i m x} e^{-2\pi i n y}}{-2\pi i m} & \text{when } m \neq 0, \\ x e^{-2\pi i n y} & \text{when } m = 0. \end{cases} \tag{16}$$

For each j, let $\{\gamma_l^j\}$ be the set of paths that form Γ_j. Along each γ_l^j we approximate the line integrals in (15) with the help of Gauss-Legendre quadrature scheme of §2.3. Indeed, using approximation (9) on each γ_l^j and substituting the result into (15) we get:

$$\hat{f}(m,n) \approx \sum_{j=1}^{J} \sum_{l=1}^{L_j} \sum_{k=1}^{q_l} K_j \cdot F_{m,n}(\chi_k^l, \psi_k^l) \cdot b \cdot \omega_k^l, \tag{17}$$

where, for each l, the points $\{(\chi_k^l, \psi_k^l)\}_{k=1}^{q_l}$ are the images under γ_l^j of the Gauss-Legendre nodes t_k of equation (9):

$$(\chi_k^l, \psi_k^l) = \gamma_l^j(t_k), \tag{18}$$

the numbers $\omega_1^l, \ldots, \omega_{q_l}^l$ are the corresponding Gauss-Legendre weights, and b has been defined in equation (14).

In order to be able to use the Fast Fourier Transform, we redistribute the weights from the Gauss-Legendre nodes in (18) to the uniform grid with the help of the Lagrange Interpolation formula (5) applied to the function $F_{m,n}$ at $(x,y) = (\chi_k^l, \psi_k^l)$ for each k and l. The final approximation to $\hat{f}(m,n)$ becomes:

$$\hat{f}(m,n) \approx \sum_{j=1}^{J} \sum_{l=1}^{L_j} \sum_{k=1}^{q_l} K_j \cdot b \cdot \omega_k^l \sum_{j_1=1}^{p} \sum_{j_2=1}^{p} \delta_{j_1}(\chi_k^l) \cdot \delta_{j_2}(\psi_k^l) \cdot F_{m,n}(x_{j_1}^k, y_{j_2}^k), \tag{19}$$

where, for each k and l, the points $\{(x_{j_1}^k, y_{j_2}^k)\}_{j_1,j_2=1}^p$ are chosen on the uniform grid so that the point (χ_k^l, ψ_k^l) lies in the h_x by h_y rectangle centered on the center of the interpolation window $[x_1^k, x_p^k] \times [y_1^k, y_p^k]$.

Remark 3.1 Changing the order of summation in (19), we see that

$$\hat{f}(m,n) \approx \sum_{j_1}\sum_{j_2}\left(\sum_{j=1}^{J}\sum_{l=1}^{L_j}\sum_{k=1}^{q_l} K_j \cdot b \cdot \omega_k^l \cdot \delta_{j_1}(\chi_k^l) \cdot \delta_{j_2}(\psi_k^l)\right) \cdot F_{m,n}(x_{j_1}, y_{j_2}), \tag{20}$$

where the points $\{(x_{j_1}, y_{j_2})\}$ form a uniform grid on $[0,1] \times [0,1]$ with sampling intervals h_x and h_y.

Let $G : \mathbb{Z}^2 \to \mathbb{C}^1$ be defined by the term in the parentheses in (20):

$$G(j_1, j_2) \stackrel{\text{def}}{=} \sum_{j=1}^{J}\sum_{l=1}^{L_j}\sum_{k=1}^{q_l} K_j \cdot b \cdot \omega_k^l \cdot \delta_{j_1}(\chi_k^l) \cdot \delta_{j_2}(\psi_k^l). \tag{21}$$

Then, substituting (21) into (20), and, using the definition of $F_{m,n}$ in (16), we get:

$$\begin{aligned} \hat{f}(m,n) &\approx \sum_{j_1}\sum_{j_2} G(j_1,j_2) \cdot F_{m,n}(x_{j_1}, y_{j_2}) \\ &= \begin{cases} \dfrac{1}{-2\pi i m}\displaystyle\sum_{j_1}\sum_{j_2} G(j_1,j_2) \cdot e^{-2\pi i m x_{j_1}} e^{-2\pi i n y_{j_2}} & \text{when } m \neq 0, \\ \displaystyle\sum_{j_2}\left(\sum_{j_1} G(j_1,j_2) \cdot x_{j_1}\right) e^{-2\pi i n y_{j_2}} & \text{when } m = 0 \end{cases} \end{aligned} \tag{22}$$

which can be readily computed for all m and n with the help of two- and one-dimensional FFTs. ■

3.3 Error Estimates

In this section we estimate the accuracy of the approximations (17) and (19) made in §3.2.

Let γ_l^j be a straight path as defined in (14):

$$\gamma_l^j(t) = (a_0 + at, b_0 + bt) \qquad \text{for } t \in [0,1].$$

Since Gauss-Legendre quadrature and Lagrange interpolation formula are translation invariant, the estimates will not change if we assume that $a_0 = b_0 = 0$. Then, for every (m,n) such that $-M < m \leq M$ and $-N < n \leq N$, using (9) and (14), we get:

$$\begin{aligned} \int_{\gamma_l^j} F_{m,n}(x,y)\,dy &= \int_0^1 F_{m,n}(at,bt) \cdot b\,dt \\ &= \sum_{k=1}^{q_l} F_{m,n}(\chi_k^l, \psi_k^l) \cdot b \cdot \omega_k^l \\ &\quad + b \cdot E_{q_l}(F_{m,n} \circ \gamma_l^j). \end{aligned} \tag{23}$$

Lemma 3.1 *For all (m,n) such that $-M < m \leq M$ and $-N < n \leq N$, the error term E_{q_l} in (23) satisfies the inequality:*

$$\left|E_{q_l}(F_{m,n} \circ \gamma_l^j)\right| \leq \frac{2^{3q_l+1}(q_l!)^4\pi^{2q_l}}{(2q_l+1)[(2q_l)!]^3}\left(\sqrt{M^2+N^2}\cdot\sqrt{a^2+b^2}\right)^{2q_l}\frac{q_l}{2\pi}. \tag{24}$$

Proof. For $\gamma_l^j : [0,1] \to \mathbb{R}^2$ as defined by (14) with $a_0 = b_0 = 0$

$$(F_{m,n} \circ \gamma_l^j)(t) = \begin{cases} \dfrac{e^{-2\pi i(ma+nb)t}}{-2\pi im} & \text{when } m \neq 0, \\ a \cdot t \cdot e^{-2\pi inbt} & \text{when } m = 0 \end{cases} \tag{25}$$

and, consequently,

$$\frac{d^{2q_l}}{d^{2q_l}t}(F_{m,n} \circ \gamma_l^j)(t) = \frac{[-2\pi i(ma+nb)]^{2q_l}\cdot e^{-2\pi i(ma+nb)t}}{-2\pi im} \tag{26}$$

when $m \neq 0$, and

$$\frac{d^{2q_l}}{d^{2q_l}t}(F_{m,n} \circ \gamma_l^j)(t) = \left[(-2\pi inb)^{2q_l} + 2q_l \cdot a \cdot (-2\pi inb)^{2q_l-1}\right] e^{-2\pi inbt} \tag{27}$$

when $m = 0$.

Let $\bar{M}$ denote $(M^2+N^2)^{1/2}$ and let $\bar{L}$ denote $(a^2+b^2)^{1/2}$. Then, it follows from (26) and (27) that for $m \neq 0$,

$$\begin{aligned} \left|\frac{d^{2q_l}}{d^{2q_l}t}(F_{m,n} \circ \gamma_l^j)(t)\right| &\leq \frac{(2\pi|ma+nb|)^{2q_l}}{2\pi|m|} \\ &\leq \frac{(2\pi\sqrt{m^2+n^2}\cdot\sqrt{a^2+b^2})^{2q_l}}{2\pi|m|} \\ &\leq \frac{(2\pi\bar{M}\bar{L})^{2q_l}}{2\pi|m|}, \end{aligned} \tag{28}$$

and, for $m = 0$,

$$\begin{aligned} \left|\frac{d^{2q_l}}{d^{2q_l}t}(F_{m,n} \circ \gamma_l^j)(t)\right| &\leq \left|(-2\pi inb)^{2q_l} + 2q_l \cdot a \cdot (-2\pi inb)^{2q_l-1}\right| \\ &\leq |2\pi nb|^{2q_l-1} \cdot |-2\pi inb + 2q_l \cdot a| \\ &\leq |2\pi nb|^{2q_l-1} \cdot \sqrt{(2\pi n)^2 + (2q_l)^2}\cdot\sqrt{a^2+b^2} \\ &\leq (2\pi\bar{M}\bar{L})^{2q_l-1}\cdot\sqrt{(2\pi\bar{M})^2+(2q_l)^2}\cdot\bar{L} \\ &\leq (2\pi\bar{M}\bar{L})^{2q_l}\cdot\sqrt{1+\left(\frac{q_l}{\pi\bar{M}}\right)^2} \\ &\leq (2\pi\bar{M}\bar{L})^{2q_l}\cdot q_l. \end{aligned} \tag{29}$$

Putting (10), (28), and (29) together, we get an estimate of the error of Gauss-Legendre quadrature in (23):

$$\left|E_{q_l}(F_{m,n}\circ\gamma_l^j)\right| \leq \begin{cases} \dfrac{2^{q_l+1}(q_l!)^4}{(2q_l+1)[(2q_l)!]^3}\cdot\dfrac{(2\pi\bar{M}\bar{L})^{2q_l}}{2\pi|m|} & \text{when } m\neq 0, \\ \dfrac{2^{q_l+1}(q_l!)^4}{(2q_l+1)[(2q_l)!]^3}\cdot(2\pi\bar{M}\bar{L})^{2q_l}\cdot q_l & \text{when } m\neq 0 \end{cases}$$

$$\leq \frac{2^{q_l+1}(q_l!)^4}{(2q_l+1)[(2q_l)!]^3}(2\pi\bar{M}\bar{L})^{2q_l}\cdot\frac{q_l}{2\pi},$$

which is exactly the claim of the lemma. ■

Corollary 3.1 *Let $\gamma : [0,1] \to \mathbb{R}^2$ be a straight path defined by (14), and let $F_{m,n}$ be a function defined by (16).*

Then, for every $\epsilon > 0$,

$$\left|\int_\gamma F_{m,n}\,dy - \sum_{k=1}^{q} F_{m,n}\left(\gamma(t_k)\right)\cdot b\cdot\omega_k\right| < \epsilon \tag{30}$$

whenever $-M < m \leq M$, $-N < n \leq N$, and

$$q > \text{const.}\cdot\max\left\{\bar{M}\bar{L}, \log\frac{1}{\epsilon}\right\}. \tag{31}$$

Here t_k and ω_k are the Gauss-Legendre nodes and weights, respectively, and $\bar{M}$, $\bar{L}$ have been defined in (28).

Proof. Using Stirling's approximation to $q!$, we see that for large q the right-hand side of expression (24) is asymptotic to:

$$\frac{2^{3q+1}(q!)^4\pi^{2q}q}{(2q+1)[(2q)!]^3 2\pi}\cdot(\bar{M}\bar{L})^{2q} \sim \frac{1}{4}\sqrt{\frac{q}{\pi}}\left(\frac{e\pi\bar{M}\bar{L}}{2^{3/2}q}\right)^{2q} \leq \left(\frac{4\bar{M}\bar{L}}{q}\right)^{2q}. \tag{32}$$

Therefore, the inequality (30) will hold if

$$q > \text{const.}\cdot\max\left\{\bar{M}\bar{L}, \log\frac{1}{\epsilon}\right\}. \tag{33}$$

■

Remark 3.2 The quantity $\bar{M}$ is the frequency of the function $F_{M,N}$ along γ, and $\bar{L}$ is the length of γ. Corollary 3.1, then, implies that the number of Gauss-Legendre nodes needed to compute the integral $\int_\gamma F_{M,N}\,dy$ with accuracy ϵ is proportional to the number of periods of $F_{M,N}$ along γ (so long as that number is greater than $\log(1/\epsilon)$). ■

After computing Gauss-Legendre nodes and weights, in the equation (19) we redistribute the weights onto the uniform grid with the help of Lagrange interpolation formula:

$$F_{m,n}(\chi_k^l,\psi_k^l) = \sum_{j_1=1}^{p}\sum_{j_2=1}^{p}\delta_{j_1}(\chi_k^l)\cdot\delta_{j_2}(\psi_k^l)\cdot F_{m,n}(x_{j_1}^k, y_{j_2}^k) + R_p(\chi_k^l,\psi_k^l), \tag{34}$$

The error $R_p(\chi_k^l,\psi_k^l)$ of this approximation is estimated in the next lemma.

Lemma 3.2 *For all (m,n) such that $-M < m \leq M$ and $-N < n \leq N$, the error term $R_p(\chi,\psi)$ satisfies the inequality:*

$$|R_p(\chi,\psi)| \leq \frac{\left[\Gamma\left(\frac{p}{2}+1\right)\right]^2}{p!} \cdot [(2\pi h_x M)^p + (2\pi h_y N)^p] \tag{35}$$

$$\leq \sqrt{\pi p} \cdot \frac{1}{2^p} \cdot [(2\pi h_x M)^p + (2\pi h_y N)^p] \tag{36}$$

for $p \geq 2$.

Proof. For $m \neq 0$, (35) follows directly from (6). When $m = 0$, we observe that the function x is approximated exactly by Lagrange interpolation formula for $p \geq 2$ (and we only consider those), so the estimate (35) applies to this case also. Inequality (36) is a consequence of Stirling's formula [1]. ■

Definition 3.1 Let $\nu = \nu(p,\epsilon)$ be the minimum number of points per wavelength that is needed to make Lagrange interpolation formula of order p approximate with accuracy ϵ all the functions

$$e^{-2\pi i m x} e^{-2\pi i n y} \qquad \text{with} \quad -M < m \leq M \quad \text{and} \quad -N < n \leq N$$

inside the central h_x by h_y rectangle of the interpolation window $[x_1, x_p] \times [y_1, y_p]$. ■

Remark 3.3 In order to assure that Lagrange interpolation has accuracy ϵ, we now choose the parameters p and ν, defined in equation (1) and Definition 3.1, respectively. In view of Definition 3.1, we set the sampling intervals h_x and h_y to be:

$$h_x = \frac{1}{\nu M} \quad \text{and} \quad h_y = \frac{1}{\nu N}. \tag{37}$$

Substituting (37) into (36), we see that

$$|R_p(\chi,\psi)| \leq \sqrt{\pi p} \cdot \frac{1}{2^p} \cdot [(2\pi h_x M)^p + (2\pi h_y N)^p] \leq \sqrt{\pi p} \cdot \left(\frac{\pi}{\nu}\right)^p. \tag{38}$$

Numerical evidence suggests that $\nu = p/2$ is a good choice. Making this choice in (38), we see that Lagrange interpolation is guaranteed to have accuracy ϵ if

$$\sqrt{\pi p} \cdot \left(\frac{2\pi}{p}\right)^p < \epsilon. \tag{39}$$

Consequently, we choose parameters p and ν of the computation according to the relations:

$$2\nu = p \sim \log \frac{1}{\epsilon}. \tag{40}$$

Numerical experiments of Section 5, however, show that the requirement (39) is unduly strict. For example, for $\epsilon = 10^{-14}$, the inequality (39) demands $p = 25$, while $p = 16$ is sufficient. Similarly, for single precision, $\epsilon = 10^{-7}$, the inequality (39) requires $p = 18$, when, in fact, $p = 10$ is large enough. ■

With the help of the lemmas 3.1 and 3.2 we can now estimate the total error of the computation.

Substituting (34) into (23), we get:

$$\int_{\gamma_l^j} F_{m,n}(x,y)\,dy = \sum_{k=1}^{q_l}\sum_{j_1=1}^{p}\sum_{j_2=1}^{p} \delta_{j_1}(\chi_k^l)\cdot\delta_{j_2}(\psi_k^l)\cdot F_{m,n}(x_{j_1}^k, y_{j_2}^k)\cdot b\cdot\omega_k^l + b\cdot E_{q_l}(F_{m,n}\circ\gamma_l^j) + \sum_{k=1}^{q_l} b\cdot\omega_k^l\cdot R_p(\chi_k^l,\psi_k^l) \tag{41}$$

Denoting by $\bar{R}_p$ the right-hand side of (36), by $\bar{E}_{q_l}$ the right-hand side of (24), and, observing that $\sum_{k=1}^{q_l}\omega_k^l = 1$, we see that the error of integrating along the path γ_l^j in (41) is bounded by

$$b\bar{E}_{q_l} + b\bar{R}_p \le 2b\epsilon \le 2\bar{L}\epsilon, \tag{42}$$

if p and ν are chosen according to relations (40) in Remark 3.3.

Summing up the errors in (42) over all paths γ_l^j that form the boundary Γ_j, we see that integration around each Γ_j results in an error not greater than

$$2\epsilon\|\Gamma_j\|,$$

where $\|\Gamma_j\|$ denotes the length of the perimeter of Γ_j. The total error of the computation is, therefore, at most

$$2\epsilon\sum_{j=1}^{J}|K_j|\cdot\|\Gamma_j\|.$$

Actual errors are reported in Section 5.

4 Formal Description of the Algorithm

4.1 Description of the Algorithm

Let $f : \mathbb{R}^2 \to \mathbb{C}^1$ be as in (13), and M and N be the highest frequencies desired in the Fourier transform of f along the x- and y-axis, respectively.

STEP 0: INITIALIZATION.

Comment: [We choose precision of the computation ϵ and determine the degree of Lagrange interpolation p and the oversampling factor ν as described in Remark 3.3. Since we will need ν points per wavelength, we create the function $G(m,n)$, defined in (21), on the uniform grid on $[0,1]\times[0,1]$ with the sampling intervals $h_x = 1/(\nu M)$ and $h_y = 1/(\nu N)$. We will also make use of a one-dimensional array $G_0(n)$ to compute the Fourier Transform for the case $m = 0$. We precompute the denominators of Lagrange interpolation and Gauss-Legendre nodes and weights on $[0,1]$.]

do $n = 1, \ldots, \nu N$
  $G_0(n) = 0$
  do $m = 1, \ldots, \nu M$
    $G(m, n) = 0$
  enddo
enddo

STEP 1: GREEN'S THEOREM AND LAGRANGE INTERPOLATION.

Comment: [For each j, we integrate the constant function K_j along each of the paths $\gamma_l^j : [0,1] \to \Gamma_j$, defined in (14), and then redistribute the weight from each of the resulting Gauss-Legendre nodes to p^2 Lagrange nodes from the uniform grid as described in equation (19). *Note: The double index* $(m(i_1), n(i_2))$ *of the array* G *corresponds to the point* (x_{i_1}, y_{i_2}).]

do $j = 1, \ldots, J$
  do $l = 1, \ldots, L_j$
    Using the precomputed nodes and weights on $[0,1]$ calculate the Gauss-Legendre nodes $\{(\chi_k^l, \psi_k^l)\}$ and weights ω_k^l (defined in (18)) needed to compute the line integrals along γ_l^j.
    do $k = 1, \ldots, q_l$
      do $i_1 = 1, \ldots, p$ *and* $i_2 = 1, \ldots, p$
        $G(m(i_1), n(i_2)) = G(m(i_1), n(i_2)) + K_j \cdot b \cdot \omega_k^l \cdot \delta_{i_1}(\chi_k^l) \cdot \delta_{i_2}(\psi_k^l)$
        $G_0(n(i_2)) = G_0(n(i_2)) + K_j \cdot \chi_k^l \cdot b \cdot \omega_k^l \cdot \delta_{i_1}(\chi_k^l) \cdot \delta_{i_2}(\psi_k^l)$
      enddo
    enddo
  enddo
enddo

STEP 2: FOURIER TRANSFORM.

Comment: [We apply the two-dimensional FFT to the νM by νN array G and only keep the frequencies (m, n) with $-M < m \leq M$ and $-N < n \leq N$. We store the result in the array FG. Similarly, we apply the one-dimensional FFT to the array G_0, keep only the frequencies n with $-N < n \leq N$, and store the result in the array FG_0.]

STEP 3: INTEGRATION.

Comment: [For each frequency (m, n) with $m \neq 0$, we divide its coefficient by $-2\pi i m$, which brings the computation in agreement with equation (19). The case $m = 0$ is already correct and needs no adjustment.]

do $m = -M + 1, \ldots, M$ *and* $n = -N + 1, \ldots, N$; *with* $m \neq 0$
  $FG(m, n) = FG(m, n)/(-2\pi i m)$
enddo
do $n = -N + 1, \ldots, N$
  $FG(0, n) = FG_0(n)$
enddo

Remark 4.1 Often, for example in VLSI modeling, the boundaries of the domains D_j contain vertical and horizontal straight-line segments. Contribution from horizontal segments to (15) is zero, and if S is a vertical segment from (x_0, y_0) to (x_0, y_1), then, for $n \neq 0$,

$$\int_S F_{m,n}(x,y)\,dy = \frac{F_{m,n}(x_0,y_1) - F_{m,n}(x_0,y_0)}{-2\pi i n}, \tag{43}$$

that is, computation of a line integral is replaced by evaluation of a sum of two terms of the same type as in the approximation (17). Effects of the resulting speed-up are discussed in Section 5. ■

4.2 Complexity Analysis of the Algorithm

Time complexity of the algorithm is summarised in Table 1. We assume in this section, for convenience, that $M = N$.

STEP 1	$p^2 N_g$	N_g is the number of Gauss-Legendre nodes created.
STEP 2	$O(\nu^2 N^2 \log N + \nu N \log N)$	One two-dimensional and one one-dimensional FFT.
STEP 3	$O(N^2)$	Division of each Fourier coefficient $FG(m,n)$ by $-2\pi i m$.

Table 1: Time complexity of the algorithm.

For a given accuracy ϵ the number of Gauss-Legendre nodes on a single path γ_l^j, according to Corollary 3.1, is

$$O\left(\max\left\{\bar{M}\bar{L}, \log\frac{1}{\epsilon}\right\}\right), \tag{44}$$

where $\bar{M}$ is the highest frequency along γ_l^j and $\bar{L}$ is the length of γ_l^j as described in Remark 3.2. Since $M = N$, $\bar{M} = N$. Therefore, the number of nodes needed for Γ_j is:

$$O\left(\max\left\{N \cdot \|\Gamma_j\|, L_j \cdot \log\frac{1}{\epsilon}\right\}\right), \tag{45}$$

where $\|\Gamma_j\|$ is the length of Γ_j, and L_j is the number of the paths γ_l^j that form Γ_j. It follows that the total number of nodes, N_g, satisfies the relation:

$$N_g = O\left(\max\left\{N \cdot \sum_{j=1}^{J} \|\Gamma_j\|, \left(\sum_{j=1}^{J} L_j\right) \cdot \log\frac{1}{\epsilon}\right\}\right). \tag{46}$$

In the worst case, i.e., when the perimeter is largest, there are $O(N^2)$ domains D_j with diameter $O(1/N)$ each. N_g, then, becomes

$$N_g = O\left(\max\left\{N \cdot N^2 \cdot \frac{1}{N}, N^2 \cdot \log\frac{1}{\epsilon}\right\}\right). \tag{47}$$

Choosing parameters p and ν according to (40), i.e., with

$$2\nu = p \sim \log \frac{1}{\epsilon}, \tag{48}$$

we arrive at a bound for time complexity of the algorithm:

$$O(p^2 N_g + \nu^2 N^2 \log N + N^2) = O\left(\left(\log^3 \frac{1}{\epsilon}\right) N^2 + \left(\log^2 \frac{1}{\epsilon}\right) N^2 \log N\right). \tag{49}$$

5 Numerical Experiments

We implemented the algorithm of section 4 both with and without the observation about the Green's theorem in §2.2. The algorithm was implemented in FORTRAN 77 and was run on SPARCstation 2. All internal calculations were performed in double-precision arithmetic, but the parameters p and ν were relaxed for single-precision experiments. Accuracy was measured in the cases when all domains were rectangles; in these cases exact analytic solution is available and it was computed in double-precision for comparison. To assure that integration along more general curves does not introduce unexpected errors, we cut the rectangles into more complicated shapes (thus preserving the final answer) and verified that the error did not increase. This suggests that the error is not higher for the regions for which closed form solutions are not available.

In the tables below we report the results of numerical experiments using the following three algorithms. In Algorithm 1 we make use of the Green's theorem as described in the preceding sections and, in addition, make use of Remark 4.1 to integrate along the horizontal and vertical lines. Algorithm 2 is also based on the Green's theorem, but without preferential treatment for horizontal or vertical lines. In Algorithm 3 the area integrals are treated directly without recourse to the Green's theorem.

When all the domains $\{D_j\}$ are rectangles, we report run times for a direct method, which we also use to estimate the errors of the computations. When, for each j, $D_j = (a_j, b_j) \times (c_j, d_j)$, the direct method consists of calculating the Fourier transform, $\phi : \mathbb{Z}^2 \to \mathbb{C}^1$, by evaluating the Fourier integrals analytically:

$$\begin{aligned} \phi(m,n) &= \sum_j \int_{a_j}^{b_j} \int_{c_j}^{d_j} e^{-2\pi i m x} e^{-2\pi i n y}\, dy dx \\ &= \sum_j \left(\frac{e^{-2\pi i m b_j} - e^{-2\pi i m a_j}}{-2\pi i m}\right) \cdot \left(\frac{e^{-2\pi i n d_j} - e^{-2\pi i n c_j}}{-2\pi i n}\right). \end{aligned}$$

The error E_∞ that we report is the maximal absolute error among all the frequencies:

$$E_\infty = \max_{\substack{-M<m\leq M \\ -N<n\leq N}} |\hat{f}(m,n) - \phi(m,n)|. \tag{50}$$

We do not report results for Algorithm 3 with $N = 256$ because its memory requirements were excessive for our implementation.

In the examples 1 and 2 we also include the performance of the standard FFT. These results are for illustrative purposes only; they can be improved with the help of some standard techniques such as Richardson extrapolation.

Example 1 Here we compute the Fourier Transform of a single large rectangle (approximately 0.6 by 0.66). Extrapolating the results for the FFT, we see that single precision accuracy, $E_\infty = 10^{-7}$, would result in time complexity of more than 10^9 operations, and for double precision we would need more than 10^{23}.

Example 2 In this and the next example we calculate the Fourier transforms of a piece of a VLSI mask (courtesy of Eytan Barouch). In simulations, these masks usually consist of a large number of simple geometric shapes that model the integrated circuit. In our case, there are 1215 rectangles and 424 triangles. In this example we compute the Fourier transform of the rectangles only in order to estimate the error of the computation. In the next example, we calculate the Fourier transform of all the domains, including the triangles. The results of this example can be found in tables 4 and 5, and those of example 3 in table 6.

Example 3 In this example we compute the Fourier Transform of 1215 small rectangles (the same ones as in the previous example) and 424 triangles. Their total area is 0.183 and the total perimeter of their union is 69.3. The total perimeter of the domains in this example (and, consequently, the run times of algorithms 1 and 2) are slightly smaller than in the previous example because some of the newly-added triangles share some of their boundaries with the rectangles. (We remove from the computation the curves whose contributions can be easily seen to cancel.)

The following observations can be made based on the numerical experiments above:

1. The errors for all three algorithms do not grow with N—the size of the problem.
2. All three algorithms have approximately the same accuracy.
3. The run times grow approximately like N^2 in example 1 for all three algorithms, and in example 2 for algorithms 1 and 3 (in agreement with (49)), while the run times for Algorithm 2 in example 2 appear to grow linearly. This apparent linear growth is the result of the large number N_g of Gauss-Legendre nodes generated, causing Lagrange interpolation to dominate the computation. Since N_g grows linearly with N, the total run times appear to grow linearly as well. The timings for Algorithm 1 in example 2 grow quadratically because FFT dominates the computation. On the other hand, in Algorithm 3 in example 2 Lagrange interpolation dominates the computation, but N_g in this case grows quadratically, so the overall growth remains quadratic. The VLSI mask of example 3 cannot be handled with the method of Remark 4.1 alone: some line integrals need to be computed numerically. As a result the run times of Algorithm 1 in example 3 grow at an intermediate rate.
4. Algorithm 2 is significantly faster than Algorithm 3, even when the ratio of perimeter to area is large. But Algorithm 1 is only twice as fast as Algorithm 2 on realistic problems (examples 2 and 3). On the same realistic problems all three algorithms are faster (for $N \geq 64$) than the closed form solution even when it is available. When the closed form solution is not available, straightforward calculations are prohibitively slow: single precision would require at least 150 years and double precision over

	Double Precision: $p = 16$ and $\nu = 8$								
	Run times (seconds)					The error E_∞			
N	Alg. 1	Alg. 2	Alg. 3	FFT	Direct	Alg. 1	Alg. 2	Alg. 3	FFT
16	2.3	3.7	129	0.1	0.2	4.8e-15	6.3e-15	6.3e-15	4.0e-02
32	11.1	14.2	671	0.2	0.8	4.6e-15	4.6e-15	3.3e-15	2.3e-02
64	52.0	57.1	2755	0.5	3.0	2.0e-15	2.0e-15	1.6e-15	9.2e-03
128	208.0	222.0	11112	2.0	12.2	1.0e-15	1.1e-15	1.1e-15	5.0e-03
256	1000.0	1028.0		8.7	48.6	1.0e-15	1.2e-15		2.5e-03

Table 2: Run times and errors for example 1; double precision.

	Single Precision: $p = 10$ and $\nu = 5$								
	Run times (seconds)					The error E_∞			
N	Alg. 1	Alg. 2	Alg. 3	FFT	Direct	Alg. 1	Alg. 2	Alg. 3	FFT
16	0.9	1.3	24	0.1	0.2	1.7e-08	1.5e-08	1.5e-08	4.0e-02
32	3.9	4.7	95	0.2	0.7	8.5e-09	8.3e-09	7.7e-09	2.3e-02
64	18.4	20.1	381	0.5	3.1	5.2e-09	4.7e-09	4.8e-09	9.2e-03
128	83.3	86.4	1538	2.0	12.2	2.0e-09	5.7e-09	4.3e-09	5.0e-03
256	339.0	345.0		8.7	48.6	1.5e-09	4.4e-09		2.5e-03

Table 3: Run times and errors for example 1; single precision.

	Double Precision: $p = 16$ and $\nu = 8$								
	Run times (seconds)					The error E_∞			
N	Alg. 1	Alg. 2	Alg. 3	FFT	Direct	Alg. 1	Alg. 2	Alg. 3	FFT
16	6.4	77	283	3.1	165	1.1e-14	1.0e-14	5.9e-15	5.7e-02
32	15.8	156	887	3.1	672	6.2e-15	9.4e-15	7.7e-15	4.2e-02
64	55.0	293	2277	3.5	2707	5.7e-15	1.1e-14	5.1e-15	3.1e-02
128	213.0	643	6562	4.8	10874	3.3e-15	7.8e-15	3.5e-15	2.9e-02
256	1004.0	1834		11.7	43740	2.4e-15	1.0e-14		1.7e-02

Table 4: Run times and errors for example 2; double precision.

	Single Precision: $p = 10$ and $\nu = 5$								
	Run times (seconds)					The error E_∞			
N	Alg. 1	Alg. 2	Alg. 3	FFT	Direct	Alg. 1	Alg. 2	Alg. 3	FFT
16	2.9	31	94	3.1	165	2.2e-08	3.8e-08	1.3e-08	5.7e-02
32	6.0	48	190	3.1	673	2.2e-08	2.0e-08	1.8e-08	4.2e-02
64	19.6	83	398	3.5	2717	1.3e-08	4.0e-08	1.3e-08	3.1e-02
128	85.2	200	1075	4.8	10920	9.2e-09	1.6e-08	9.0e-09	2.9e-02
256	338.0	562		11.7	43630	5.3e-09	2.7e-08		1.7e-02

Table 5: Run times and errors for example 2; single precision.

	Double precision: $p = 16$ and $\nu = 8$			Single precision: $p = 10$ and $\nu = 5$		
N	Alg. 1	Alg. 2	Alg. 3	Alg. 1	Alg. 2	Alg. 3
16	11	73	333	4.2	29	107
32	23	148	989	8.1	45	216
64	65	282	2451	22.3	80	441
128	231	628	6972	89.7	193	1153
256	1036	1804		347.0	555	

Table 6: Run times (in seconds) for example 3.

10^{16} years regardless of the values of N. Adjustments such as repeated Richardson extrapolation will undoubtedly improve these times, but are unlikely to make them competitive.

6 Conclusions and Generalizations

6.1 Generalizations

The entire algorithm of Section 4, including Remark 4.1, readily generalizes to higher dimensions; we will not describe the details of the higher-dimensional implementation here.

We will describe an extension of the algorithm to piecewise smooth functions. If the function $f(x,y)$ in (13) is piecewise smooth, i.e., for each j, the coefficient $K_j = K_j(x,y)$ is a smooth function of (x,y), we analyze it into a Fourier series with respect to x:

$$K_j(x,y) = \sum_m \hat{K}_j(m)(y) e^{2\pi i m x}, \tag{51}$$

where

$$\hat{K}_j(m)(y) \stackrel{\text{def}}{=} \int_0^1 K_j(x,y) e^{-2\pi i m x}\, dx.$$

We observe next that all the computations of §3.2 continue to hold if K_j is a function of y alone. That is, the algorithm of Section 4 can be applied to $\hat{K}_j(m)(y)$ separately for each m and the results added up afterwards. The time complexity of this new algorithm would be that of the algorithm of Section 4 times the width of the widest Fourier transform (in the x-direction only) of the functions $K_j(x,y)$.

6.2 Conclusions

We have presented an algorithm for evaluation of the Fourier transform of piecewise constant functions of two variables. The algorithm overcomes the accuracy problems associated with computing the Fourier transform of discontinuous functions; in fact, its time complexity is $O(\log^3(1/\epsilon)N^2 \log N)$, where ϵ is the accuracy and N is the size of the problem. The algorithm is based on the Lagrange interpolation formula and the Green's theorem, which are used to preprocess the data before applying the FFT. It admits natural generalizations to higher dimensions and to piecewise smooth functions.

References

[1] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*, (Dover, New York, 1970).

[2] E. Oran Brigham, *The Fast Fourier Transform and its Applications*, (Prentice Hall, Englewood Cliffs, NJ, 1988).

[3] J. W. Cooley and J. W. Tukey, An algorithm for the machine computation of complex Fourier series, *Math. Comp.* **19**, (1965).

[4] G. Dahlquist and Å. Björck, *Numerical Methods*, (Prentice-Hall, Englewood Cliffs, NJ, 1974).

[5] D. Gottlieb, M. Y. Hussaini, and S. Orszag, in *Spectral Methods for Partial Differential Equations*, edited by R. G. Voigt, D. Gottlieb, and M. Y. Hussaini. (SIAM, Philadelphia PA, 1984).

[6] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, (Academic Press, New York, 1980).

[7] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, (Springer Verlag, New York, 1980).

[8] H. Joseph Weaver, *Theory of Discrete and Continuous Fourier Analysis*, (Wiley, 1989).