AN ALGORITHM FOR THE CONVOLUTION OF LEGENDRE SERIES

NICHOLAS HALE* AND ALEX TOWNSEND†

Abstract. An $\mathcal{O}(N^2)$ algorithm for the convolution of compactly supported Legendre series is described. The algorithm is derived from the convolution theorem for Legendre polynomials and the recurrence relation satisfied by spherical Bessel functions. Combining with previous work yields an $\mathcal{O}(N^2)$ algorithm for the convolution of Chebyshev series. Numerical results are presented to demonstrate the improved efficiency over the existing algorithm.

Key words. convolution, Legendre polynomial, spherical Bessel function, Fourier transform

AMS subject classifications. 33C10, 33C45, 44A35

1. Introduction. Convolution is a fundamental operation which arises in many fields, particularly in signal processing [11,14] and statistics [8]. Given two continuous integrable functions, f and g, their convolution is a third function, h, defined formally by the integral

$$h(x) = (f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)\,dt, \qquad x \in \mathbb{R}, \tag{1.1}$$

which represents the area of overlap when g is reversed and shifted over f. This operation is typically denoted by an asterisk '$*$' and is commutative and distributive.

In this article we focus on the particular case, which we shall motivate shortly, where f and g are polynomials of finite degree with zero support outside of $[a, b]$ and $[c, d]$, respectively. In this case, the *convolution domain* is the region in the (x, t)-plane where both f and g are non-zero (see Figure 1.1), which gives rise to the following equivalent definition:

$$h(x) = (f * g)(x) = \int_{\max(a,x-d)}^{\min(b,x-c)} f(t)g(x-t)\,dt, \qquad x \in [a+c, b+d], \tag{1.2}$$

and $h(x) = 0$ for $x \notin [a + c, b + d]$.

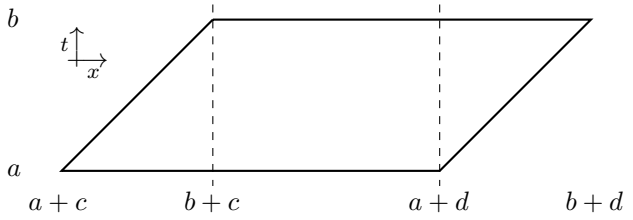

Fig. 1.1: The convolution domain when f and g have support on $[a, b]$ and $[c, d]$, respectively. Throughout we assume, without loss of generality, that $d - c \geq b - a$.

*Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK, (hale@maths.ox.ac.uk, http://people.maths.ox.ac.uk/hale/). Work was supported by The MathWorks, Inc., and King Abdullah University of Science and Technology (KAUST), award KUK-C1-013-04.

†Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK, (townsend@maths.ox.ac.uk, http://people.maths.ox.ac.uk/townsend/). Work was supported by EPSRC grant EP/P505666/1 and by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement 291068.

Existing algorithms for computing (1.1) are usually based on the assumption that f and g are periodic functions, which allows use of the Fourier transform and the *convolution theorem* [10, p. 6]

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}, \qquad \mathcal{F}\{f\}(\omega) = \int_{-\infty}^{\infty} f(x) e^{\mathrm{i}x\omega}\, dx.$$

When f and g have the same period, $f*g$ can be computed in $\mathcal{O}((M+N)\log(M+N))$ operations by approximating them by finite Fourier expansions of length M and N, respectively, and by employing the FFT [10, sec. 6.2.4]. However, this does not allow f and g to have different periods or compact support.

A powerful computing paradigm for working with continuous functions f and g on bounded intervals is to replace them by polynomial approximants (or "proxies") f_M and g_N of sufficiently high degree so that $\|f - f_M\|_\infty$ and $\|g - g_N\|_\infty$ are on the order of machine precision [3,9,16]. Properties of f and g can then be efficiently calculated, up to an error of machine precision, by numerically computing with the polynomials f_M and g_N. Until now, one major exception was the computation of convolutions, with the best known algorithm being an approach that evaluates (1.2) directly using a quadrature rule (see section 2), which required $\mathcal{O}((M+N)^3)$ operations.

In this article we describe a new algorithm that takes as its starting point Legendre series representations of the polynomials f_M and g_N that makes use of the Fourier transform of Legendre polynomials (see (3.1)) and the recurrence relation satisfied by spherical Bessel functions (see (4.4)), to ultimately compute the corresponding Legendre series coefficients of h. This new algorithm requires just $\mathcal{O}((M+N)^2)$ operations.

In the next section we introduce certain properties of h that follow from f_M and g_N being polynomials and describe the $\mathcal{O}((M+N)^3)$ quadrature-based algorithm. In section 3 we describe the Fourier transform of a compactly supported Legendre series[1] and demonstrate that the convolution theorem holds for compactly supported Legendre polynomials. Section 4 describes the new $\mathcal{O}((M+N)^2)$ algorithm when f_M and g_N have support only on $[-1,1]$, and section 5 extends this approach to more general intervals. Numerical examples which demonstrate the computational speed of our proposed algorithm are presented in section 6, before we finish with some further implementation details and concluding remarks.

An implementation of the algorithm is publicly available in Chebfun [16], an open-source software package written in object-oriented MATLAB. All numerical experiments in this paper use the `conv` command in Chebfun.

2. Quadrature-based algorithm. The quadrature-based algorithm relies on the observation that $h = f_M * g_N$ is a piecewise polynomial of two or three pieces.

LEMMA 2.1. *If f_M and g_N are polynomials of degree M and N supported on $[a,b]$ and $[c,d]$, respectively, with $d-c \geq b-a$, then $h = f_M * g_N$ is a piecewise polynomial on $[a+c, b+d]$ such that*

$$h(x) = (f_M * g_N)(x) = \begin{cases} h^{\text{left}}(x) \in \mathbb{P}^{M+N+1}, & x \in [a+c, b+c], \\ h^{\text{mid}}(x) \in \mathbb{P}^{N}, & x \in [b+c, a+d], \\ h^{\text{right}}(x) \in \mathbb{P}^{M+N+1}, & x \in [a+d, b+d]. \end{cases}$$

[1] Throughout this paper, the phrase "Legendre polynomial" is used to describe the function which coincides with the standard Legendre polynomial on $[-1,1]$, and is zero elsewhere. A "Legendre series" is a finite linear combination of such polynomials. See section 3 for further discussion.

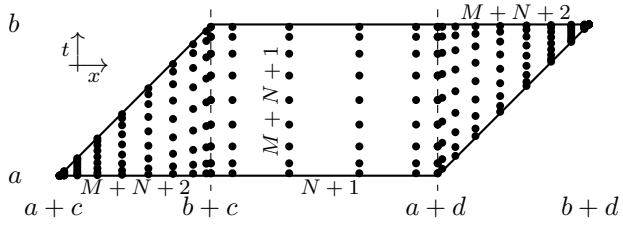

Fig. 2.1: The quadrature nodes required by the quadrature-based algorithm for $M = 3$ and $N = 5$. The number of Gauss–Legendre quadrature nodes in t (vertical direction) is chosen to exactly integrate (1.2), and the number of evaluation points in x (horizontal direction) is chosen to uniquely determine h by polynomial interpolation (see Lemma 2.1).

Proof. It is enough to consider $x^M * x^N$ on each interval. Substituting this into (1.2) and using the binomial theorem to expand $(x - t)^N$ reveals the required result. □

Therefore, the piecewise polynomial h is made of two pieces if $b + c = a + d$ (or equivalently $d - c = b - a$), and three pieces otherwise. On each piece it can be evaluated exactly at any point by a polynomial-based quadrature rule of sufficiently high degree. The uniqueness of polynomial interpolation ensures that h^{left} and h^{right} can be recovered from their values at $M + N + 2$ unique points in $[a + c, b + c]$ and $[a + d, b + d]$, respectively, and h^{mid} from $N + 1$ points in $[b + c, a + d]$. In practice, for stability and efficiency, these evaluation points are typically chosen to be Chebyshev grids on each interval. The Legendre coefficients of h can be readily recovered from such grids [2, 7].

For each fixed evaluation point $x \in [a + c, b + d]$, a polynomial-based quadrature rule of sufficient degree can then be used to compute the resulting definite integral in (1.2) exactly. In particular, an $\lceil (M + N + 1)/2 \rceil$-point Gauss–Legendre rule or an $(M + N + 1)$-point Clenshaw–Curtis rule will suffice. Figure 2.1 shows these quadrature nodes required for the case $M = 3$ and $N = 5$ when using a Gauss–Legendre quadrature rule.

This approach is accurate and stable, but expensive. Since values of f_M and g_N are required at $\mathcal{O}((M + N)^2)$ points, and evaluating a polynomial of degree N at an arbitrary point requires $\mathcal{O}(N)$ operations, the total complexity is $\mathcal{O}((M + N)^3)$.

3. The Fourier transform and the convolution theorem for Legendre polynomials. The Legendre polynomials, denoted by $P_0, P_1, \ldots$, are orthogonal on $[-1, 1]$ with respect to the weight $w(x) = 1$. They can be defined recursively, and satisfy a 3-term recurrence relation [13, (18.9.1)]

$$(n + 1)P_{n+1}(x) = (2n + 1)xP_n(x) - nP_{n-1}(x), \quad n \geq 1, \quad P_1(x) = x, \quad P_0(x) = 1.$$

In the special case of Legendre polynomials defined as functions on $[-1, 1]$ the Fourier transform can be expressed in terms of a spherical Bessel function.

THEOREM 3.1 (Fourier Transform of a Legendre polynomial). *Let $m > 0$ be an integer and P_m be the Legendre polynomial of degree m. Then*

$$\mathcal{F}\{P_m\} = \int_{-1}^{1} P_m(x)e^{-\mathrm{i}\omega x}\,dx = 2(-\mathrm{i})^n j_n(\omega), \qquad \omega \in \mathbb{R}, \tag{3.1}$$

where $j_n(z)$ is the spherical Bessel function with parameter n.

Proof. Combining [13, (18.17.19)] and [1, (10.1.1)], we have

$$\int_{-1}^{1} P_m(x)e^{-\mathrm{i}\omega x}\,dx = \mathrm{i}^n\sqrt{\frac{2\pi}{-\omega}}J_{n+\frac{1}{2}}(-\omega) = 2\mathrm{i}^n j_n(-\omega).$$

The result follows from $j_n(-\omega) = (-1)^n j_n(\omega)$ [13, (10.47.14)]. □

Moveover, if $f : [a, b] \to \mathbb{C}$ is a continuous function and $\hat{f}$ denotes its Fourier transform, we can define the inverse Fourier transform in the usual way,

$$\mathcal{F}^{-1}\{\hat{f}\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(\omega)e^{\mathrm{i}\omega x}d\omega, \qquad x \in [a, b],$$

and observe that the *Fourier inversion theorem* holds, i.e., $\mathcal{F}^{-1}\{\mathcal{F}\{f\}\} = f$.

The *convolution theorem* states that the Fourier transform of the convolution of two functions is also the pointwise product of the Fourier transforms of the two functions.

THEOREM 3.2 (Convolution theorem). *Let $f, g : \mathbb{R} \to \mathbb{C}$ be integrable continuous functions. Then*

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}, \qquad \mathcal{F}\{f\} = \int_{-\infty}^{\infty} f(x)e^{-\mathrm{i}\omega x}\,dx, \quad \omega \in \mathbb{R}.$$

Proof. This theorem appears in many textbooks. See, for example, [10, p. 6]. □

To extend this result to functions supported on $[a, b]$ we can define them to be zero off their support and hence, if $f : [a, b] \to \mathbb{C}$ is continuous (and therefore integrable), then

$$\mathcal{F}\{f\} = \int_{a}^{b} f(x)e^{-\mathrm{i}\omega x}\,dx, \qquad \omega \in \mathbb{R}.$$

By combining Theorem 3.1, Theorem 3.2, and the Fourier inversion theorem the convolution of two Legendre polynomials can be expressed as the inverse Fourier transform of a product of spherical Bessel functions.

COROLLARY 3.3 (Convolution of Legendre polynomials). *For integers $m, n > 0$,*

$$(P_m * P_n)(x) = \frac{2(-\mathrm{i})^{m+n}}{\pi}\int_{-\infty}^{\infty} j_m(\omega)j_n(\omega)e^{\mathrm{i}\omega x}\,d\omega, \qquad x \in [-2, 2]. \tag{3.2}$$

Proof. Let $h = P_m * P_n$ where P_m and P_n are defined as functions on $[-1, 1]$. By Theorems 3.1 and 3.2 we have

$$\mathcal{F}\{h\}(\omega) = \int_{-2}^{2} h(x)e^{-\mathrm{i}\omega x}\,dx = 4(-\mathrm{i})^{m+n}j_m(\omega)j_n(\omega), \qquad \omega \in \mathbb{R},$$

and by the Fourier inversion theorem we conclude that, for $x \in [-2, 2]$,

$$h(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathcal{F}\{h\}(\omega)e^{\mathrm{i}\omega x}\,d\omega = \frac{2(-\mathrm{i})^{m+n}}{\pi}\int_{\infty}^{\infty} j_m(\omega)j_n(\omega)e^{\mathrm{i}\omega x}\,d\omega,$$

as required. □

At first glance it is not obvious why Corollary 3.3 is useful for computing convolutions involving Legendre polynomials since (3.2) is a devilish infinite integral with an oscillatory and slowly decaying integrand. However, as we shall show shortly, (3.2) can be used implicitly to derive a useful recurrence relation (see Theorem 4.1). In particular, our algorithm does not explicitly work with spherical Bessel functions at any point, even though our mathematical derivations in the next section exploit their properties.

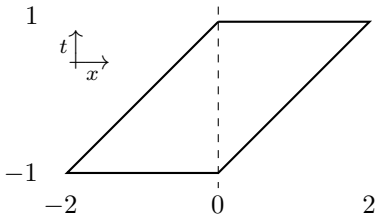

Fig. 4.1: The convolution domain for two Legendre series on $[-1,1]$.

4. Convolution of Legendre series defined on intervals of the same length. In this section we describe the convolution of two Legendre series on $[-1,1]$, before generalizing to intervals $[a,b]$ and $[c,d]$ where $d-c=b-a$.

Let f_M and g_N be two Legendre series on $[-1,1]$ of degree M and N with coefficients $\alpha_0,\ldots,\alpha_M$ and $\beta_0,\ldots,\beta_N$, respectively, such that

$$f_M(x)=\sum_{m=0}^{M}\alpha_m P_m(x),\qquad g_N(x)=\sum_{n=0}^{N}\beta_n P_n(x). \tag{4.1}$$

When $[a,b]=[c,d]=[-1,1]$ the convolution in (1.2) becomes

$$h(x)=(f_M * g_N)(x)=\int_{\max(-1,x-1)}^{\min(1,x+1)} f_M(t)g_N(x-t)\,dt,\qquad x\in[-2,2], \tag{4.2}$$

and Figure 4.1 shows the convolution domain in this case.

Here the piecewise polynomial, h, is made of two pieces, h^{left} on $[-2,0]$ and h^{right} on $[0,2]$, each of degree $N+M+1$. We construct h^{left} and h^{right} by computing their Legendre coefficients. We focus on h^{left} since the computation for h^{right} is similar.

Let $\gamma_0^{\text{left}},\ldots,\gamma_{M+N+1}^{\text{left}}$ be the Legendre coefficients of h^{left}, so that

$$h^{\text{left}}(x)=\int_{-1}^{x+1} f_M(t)g_N(x-t)\,dt=\sum_{k=0}^{M+N+1}\gamma_k^{\text{left}}P_k(x+1),\qquad x\in[-2,0].$$

By the orthogonality of Legendre polynomials and the orthonormalization constant $(k+1/2)^{-1/2}$ for $P_k(x)$ for $k=0,\ldots,M+N+1$, we have

$$\begin{aligned}\gamma_k^{\text{left}}&=\frac{2k+1}{2}\int_{-2}^{0}P_k(x+1)\int_{-1}^{x+1}f_M(t)g_N(x-t)\,dt\,dx\\&=\sum_{n=0}^{N}\beta_n\underbrace{\left[\frac{2k+1}{2}\sum_{m=0}^{M}\alpha_m\int_{-2}^{0}P_k(x+1)\int_{-1}^{x+1}P_m(t)P_n(x-t)\,dt\,dx\right]}_{=B_{k,n}^{\text{left}}}.\end{aligned} \tag{4.3}$$

The relations in (4.3) can therefore be written in matrix form as $\underline{\gamma}^{\text{left}}=B^{\text{left}}\underline{\beta}$.

We now seek an efficient way to form the matrix B^{left}. Computing the integrals in (4.3) via quadrature would be as expensive, if not more so, than the algorithm described in section 2. Instead, we exploit the fact that the entries of the B^{left} satisfy a recurrence relation, which is a consequence of the 3-term recurrence satisfied by spherical Bessel functions [13, (10.51.1)]:

$$j_{n+1}(z)=\frac{2n+1}{z}j_n(z)-j_{n-1}(z),\quad n\geq 0,\quad j_0(z)=\frac{\sin z}{z},\quad j_{-1}(z)=\frac{\cos z}{z}. \tag{4.4}$$

THEOREM 4.1. *Let M be an integer, $\alpha_0, \ldots, \alpha_M \in \mathbb{C}$, and the matrix B^{left} be defined entry-wise as*

$$B^{\text{left}}_{k,n} = \frac{2k+1}{2} \sum_{m=0}^{M} \alpha_m \int_{-2}^{0} P_k(x+1) \int_{-1}^{x+1} P_m(t) P_n(x-t)\, dt\, dx, \tag{4.5}$$

where $0 \le k \le M+N+1$ and $0 \le n \le N$. The entries of B^{left} satisfy the following recurrence relation:

$$B^{\text{left}}_{k,n+1} = -\frac{2n+1}{2k+3} B^{\text{left}}_{k+1,n} + \frac{2n+1}{2k-1} B^{\text{left}}_{k-1,n} + B^{\text{left}}_{k,n-1}, \qquad n,k \ge 1. \tag{4.6}$$

Proof. We first use the definition (4.2) for $x \in [-2,0]$ and apply the change variables $s = x+1$ to obtain

$$B^{\text{left}}_{k,n} = \frac{2k+1}{2} \sum_{m=0}^{M} \alpha_m \int_{-1}^{1} P_k(s)\, (P_m * P_n)\, (s-1)\, ds.$$

By Corollary 3.3 we have,

$$B^{\text{left}}_{k,n} = \frac{2k+1}{\pi} \sum_{m=0}^{M} (-\mathrm{i})^{m+n} \alpha_m \int_{-1}^{1} P_k(s) \int_{-\infty}^{\infty} j_m(\omega) j_n(\omega) e^{\mathrm{i}\omega(s-1)}\, d\omega\, ds.$$

Next, we swap the order of integration and use Theorem 3.1 to write the entries of B^{left} as integrals involving the triple-product of spherical Bessel functions

$$B^{\text{left}}_{k,n} = \frac{2(2k+1)}{\pi} \sum_{m=0}^{M} (-\mathrm{i})^{m+n} \mathrm{i}^k \alpha_m \int_{-\infty}^{\infty} j_k(\omega) j_m(\omega) j_n(\omega) e^{-\mathrm{i}\omega}\, d\omega. \tag{4.7}$$

Finally, using the 3-term recurrence (4.4) we have, for $k,n \ge 1$,

$$\begin{aligned} B^{\text{left}}_{k,n+1} &= \frac{2(2k+1)}{\pi} \sum_{m=0}^{M} (-\mathrm{i})^{m+n+1} \mathrm{i}^k \alpha_m \int_{-\infty}^{\infty} j_k(\omega) j_m(\omega) j_{n+1}(\omega) e^{-\mathrm{i}\omega}\, d\omega \\ &= \frac{2(2k+1)(2n+1)}{\pi} \sum_{m=0}^{M} (-\mathrm{i})^{m+n+1} \mathrm{i}^k \alpha_m \int_{-\infty}^{\infty} \frac{j_k(\omega)}{\omega} j_m(\omega) j_n(\omega) e^{-\mathrm{i}\omega}\, d\omega + B^{\text{left}}_{k,n-1} \\ &= -\frac{2n+1}{2k+3} B^{\text{left}}_{k+1,n} + \frac{2n+1}{2k-1} B^{\text{left}}_{k-1,n} + B^{\text{left}}_{k,n-1}, \end{aligned}$$

as required. □

In practice we observe that the recurrence relation (4.6) is numerically stable for computing entries of B^{left} below the diagonal, but unstable above it. This is perhaps to be expected, as when $k < n$ the factors $(2n+1)/(2k+3)$ and $(2n+1)/(2k-1)$ in (4.6) are larger than 1, and rounding errors accumulate in subtracting consecutive terms. Below the diagonal, where $k > n$, the terms are decreasing in magnitude and the recurrence is stable. The instability above the diagonal is easily overcome by observing that the matrix B^{left} is symmetric up to a scaling factor. In particular, from (4.7) it can be readily seen that

$$B^{\text{left}}_{k,n} = (-1)^{n+k} \frac{2k+1}{2n+1} B^{\text{left}}_{n,k}, \qquad n,k \ge 0, \tag{4.8}$$

which can be used to compute the otherwise unstable entries.

To initialize the recurrence in (4.6) we require the first two columns of B^{left}. Substituting $P_0(x-t)=1$ into (4.5) we find that the first column simply contains the coefficients of the indefinite integral of f_M [5, eq. 22],

$$B^{\text{left}}_{k,0} = \frac{2k+1}{2}\int_{-1}^{1} P_k(x)\int_{-1}^{x} f_M(t)\,dt\,dx = \begin{cases} \frac{\alpha_{k-1}}{2k-1} - \frac{\alpha_{k+1}}{2k+3}, & k \neq 0, \\ \alpha_0 - \alpha_1/3, & k = 0. \end{cases}$$

Similarly, the second column can be computed by observing that

$$\begin{aligned} B^{\text{left}}_{k,1} &= \frac{2k+1}{2}\int_{-1}^{1} P_k(x)\int_{-1}^{x} f_M(t)(x-t-1)\,dt\,dx \\ &= \frac{2k+1}{2}\int_{-1}^{1} P_k(x)\int_{-1}^{x} f_M(t)\int_{t}^{x} ds\,dt\,dx - B^{\text{left}}_{k,0} \\ &= \frac{2k+1}{2}\int_{-1}^{1} P_k(x)\int_{-1}^{x}\int_{-1}^{s} f_M(t)\,dt\,ds\,dx - B^{\text{left}}_{k,0}, \end{aligned}$$

where the last equality comes from interchanging the order of integration. Therefore, $B^{\text{left}}_{k,1}$ is the difference between the Legendre coefficients resulting from two consecutive indefinite integrals of f_M and $B^{\text{left}}_{:,0}$, and so, for $0 \le k \le M+N$,

$$B^{\text{left}}_{k,1} = \begin{cases} B^{\text{left}}_{k-1,0}/(2k-1) - B^{\text{left}}_{k,0} - B^{\text{left}}_{k+1,0}/(2k+3), & k \neq 0, \\ -B^{\text{left}}_{1,0}/3, & k = 0. \end{cases}$$

We suspect it may be possible to continue in this way and derive the recurrence relation (4.6) for the remaining columns of B^{left} without all the machinery of section 3, however we were unable to do so. Furthermore, it is not clear that such a derivation would reveal the symmetry relation (4.8), which is necessary to compute the terms above the diagonal in a numerically stable way.

We have shown that both $B^{\text{left}}_{:,0}$ and $B^{\text{left}}_{:,1}$ can be computed in $\mathcal{O}(M+N)$ operations, and the whole $(M+N)\times N$ matrix, B^{left}, in $\mathcal{O}((M+N)N)$ operations. The matrix-vector product $B^{\text{left}}\underline{\beta}$ can be computed with the same cost, and therefore the coefficients $\underline{\gamma}^{\text{left}}$ of h^{left} in $\mathcal{O}((M+N)N)$ operations. The coefficients $\underline{\gamma}^{\text{right}}$ of h^{right} can be computed from $B^{\text{right}}\underline{\beta}$, for which an almost identical recurrence relation can be derived.

One immediate extension is to the convolution of two Legendre series defined on intervals $[a,b]$ and $[c,d]$ of the same length, i.e., $d-c=b-a$. A Legendre polynomial on $[a,b]$ is defined as the composition $P_k \circ \psi_{[a,b]}$, where $\psi_{[a,b]}(x) = 2(x-a)/(b-a)-1$ is the linear map from $[a,b]$ to $[-1,1]$. Moreover, Legendre series of f_M and g_N on $[a,b]$ and $[c,d]$, respectively, are defined as

$$f_M(x) = \sum_{m=0}^{M} \alpha_m (P_m \circ \psi_{[a,b]})(x), \qquad g_N(x) = \sum_{n=0}^{N} \beta_n (P_n \circ \psi_{[c,d]})(x). \tag{4.9}$$

The following lemma relates $f_M * g_N$ to a convolution of series defined on $[-1,1]$.

LEMMA 4.2. *Let f and g be continuous functions defined on $[a,b]$ and $[c,d]$, respectively, with $d-c=b-a$. Then*

$$(f*g)(x) := \int_{\max(a,x-c)}^{\min(b,x-d)} f(t)g(x-t)\,dt = \frac{b-a}{2}\left((f\circ\psi^{-1}_{[a,b]}) * (g\circ\psi^{-1}_{[c,d]})\right)(y),$$

where $x \in [a+c, b+d]$ *and* $y = 2\psi_{[a+c,b+d]}(x) \in [-2, 2]$.

Proof. By the changes of variables $s = \psi_{[a,b]}(t)$ and $y = 2\psi_{[a+c,b+d]}(x)$ we have

$$\begin{aligned}(f * g)(x) &= \frac{b-a}{2} \int_{\max(-1,y+1)}^{\min(1,y-1)} f\left(\psi_{[a,b]}^{-1}(s)\right) g\left(\psi_{[a+c,b+d]}^{-1}(y/2) - \psi_{[a,b]}^{-1}(s)\right) ds \\ &= \frac{b-a}{2}\left((f \circ \psi_{[a,b]}^{-1}) * (g \circ \psi_{[c,d]}^{-1})\right)(y),\end{aligned}$$

where the last equality uses $\psi_{[c,d]}^{-1}(y-s) = \psi_{[a+c,b+d]}^{-1}(y/2) - \psi_{[a,b]}^{-1}(s)$. ∎

5. Convolution of Legendre series supported on general intervals. In this section we relax the requirement that f_M and g_M are defined on intervals of the same length. We consider three cases in turn;

i $1 < (d-c)/(b-a)$ is an integer (Figure 5.1),
ii $1 < (d-c)/(b-a) < 2$ (Figure 5.2),
iii $(d-c)/(b-a) > 2$ (Figure 5.3).

(i) $(\mathbf{d-c})/(\mathbf{b-a}) > \mathbf{1}$ **is an integer.** If $r = (d-c)/(b-a) > 1$ is an integer, g_N can be partitioned into a sum of r functions,

$$g_N(x) = \sum_{j=1}^{r} g_N^{[j]}(x), \qquad g_N^{[j]}(x) = g_N \chi_{[c+(j-1)(b-a),\, c+j(b-a)]},$$

where $\chi_{[a,b]}$ is the indicator function on the interval $[a,b]$ and each $g_N^{[j]}$, $1 \le j \le r$, is defined on an interval of length $b-a$. Therefore, by distributivity of '$*$', $f_M * g_N$ can be computed with r convolutions,

$$h = (f_M * g_N) = \sum_{j=1}^{r} (f_M * g_N^{[j]}), \tag{5.1}$$

where each convolution in the sum involves functions defined on intervals of the same length, which can be computed by the algorithm described in section 4.

For (5.1) we must *restrict* g_N to r intervals of length $b-a$ and compute the Legendre coefficients of $g_N^{[1]}, \ldots, g_N^{[r]}$. Noting that the $g_N^{[j]}$ are polynomials of degree at most N, for each j we evaluate g_N at $N+1$ Chebyshev points on $[c+(j-1)(b-a), c+j(b-a)]$ using Clenshaw's algorithm for Legendre series [4], and then compute the Chebyshev coefficients of $g_N^{[j]}$ using a discrete Chebyshev transform [6]. From these, the Legendre coefficients of $g_N^{[j]}$ can be computed using a fast Chebyshev–Legendre transform [2, 7].

Previously, we saw that the function h on $[a+c, b+d]$ is a piecewise polynomial made up of three pieces h^{left}, h^{mid}, and h^{right}. The Legendre coefficients for h^{left} and h^{right} are simply the left polynomial part of the first and the right polynomial part of the last term in (5.1). By Lemma 2.1 we know that h^{mid} is a polynomial of degree at most N, and thus is uniquely determined by it's values at $N+1$ Chebyshev points on $[b+c, a+d]$. The evaluation of h^{mid} for a given point requires summing the contribution from the terms in the righthand side of (5.1), only two of which will be non-zero. Since each of these terms are convolutions on intervals of the same length, they can be computed as described in section 4, and the evaluations can be performed using Clenshaw's algorithm for Legendre series, as described above.

Figure 5.1 shows the convolution domain for the case when $(d-c)/(b-a) = 3$. The crosses indicate the evaluation points in $x \in [b+c, a+d]$ used to construct the Legendre series for h^{mid} when $N = 7$.

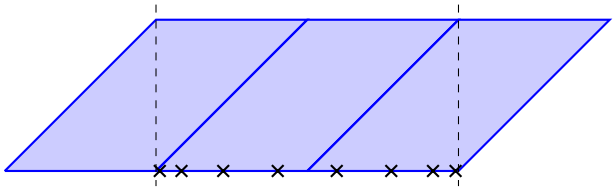

Fig. 5.1: The convolution domain when $d-c$ is an integer multiple of $b-a$. Here, $d-c=3(b-a)$. The diagram shows that the region can be subdivided into $(d-c)/(b-a)$ subdomains, each one involving a convolution of two Legendre series defined on intervals of the same length. The crosses indicate the evaluation points used to compute the Legendre coefficients for h^{mid} when $N=7$.

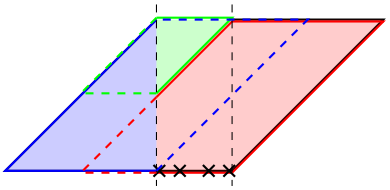

Fig. 5.2: The convolution domain when $1<(d-c)/(b-a)<2$ is divided into three subdomains, for each of which the convolution can be computed using the algorithm in section 4. The polynomial h^{left} is computed by the left-most triangle (blue), h^{right} by the right-most triangle (red), and h^{mid} by evaluating at $N+1$ Chebyshev points, summing up the contribution from the top-most triangle (green) and trapezium (red). The crosses are the evaluation points used for h^{mid} when $N=3$.

(ii) $1<(\mathbf{d}-\mathbf{c})/(\mathbf{b}-\mathbf{a})<2$. Suppose that $1<(d-c)/(b-a)<2$, so that the interval $[c,d]$ cannot be partitioned into intervals of length $b-a$. In this case, both f_M and g_N must be restricted and it can be shown that

$$h=\begin{cases}\left(f_M * g_N\chi_{[c,c+b-a]}\right), & \text{on } [a+c,b+c],\\ \left(f_M\chi_{[2b+c-d-a,b]} * g_N\chi_{[a+c,d-b+2a]}\right)+\left(f_M * g_N\chi_{[d-b+a,d]}\right), & \text{on } [b+c,a+d],\\ \left(f_M * g_N\chi_{[d-b+a,d]}\right), & \text{on } [a+d,b+d].\end{cases}$$

Therefore, for each $x\in[a+c,b+d]$, $h(x)$ can be expressed by convolutions involving functions defined on intervals of the same length. The Legendre series f_M and g_N are restricted to a desired subinterval by evaluating at $M+1$ and $N+1$ Chebyshev points, respectively, and converting the values to the corresponding Legendre coefficients, as described above. The polynomial h^{mid} is then constructed by evaluating at $N+1$ Chebyshev points in $[b+c,a+d]$.

Figure 5.2 shows the convolution domain for $1<(d-c)/(b-a)<2$. The domain is divided into 3 subdomains, each of which can be treated by the algorithm described in section 4. The crosses indicate the evaluation points in $x\in[b+c,a+d]$ used to compute the Legendre coefficients for h^{mid} when $N=3$.

(iii) $(\mathbf{d}-\mathbf{c})/(\mathbf{b}-\mathbf{a})>2$. For the final case, suppose that $(d-c)/(b-a)>2$, so that the interval $[c,d]$ can be divided into intervals of length $b-a$ plus a remainder. Let $r=\lfloor(d-c)/(b-a)\rfloor$ and write, using the distributivity of '$*$',

$$(f_M * g_N)=\left(f_M * g_N\chi_{[c,c+(r-1)(b-a)]}\right)+\left(f_M * g_N\chi_{[c+(r-1)(b-a),d]}\right). \tag{5.2}$$

The first term in (5.2) satisfies case (i) and the second satisfies case (ii), and so this final case can be reduced to the previous two. As before, h^{mid} is evaluated at $N+1$ Chebyshev points in $[b+c,a+d]$ and these values are converted to Legendre coefficients using fast transforms. Figure 5.3 shows the convolution domain when

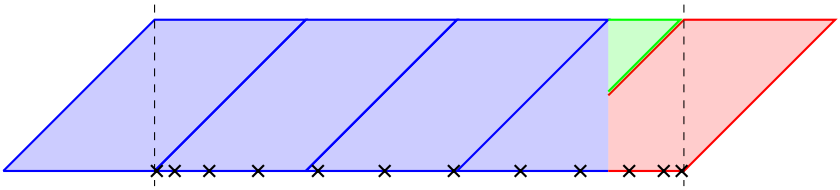

Fig. 5.3: The convolution domain for general intervals with $d-c > b-a$. Here, $4 < (d-a)/(b-a) < 5$. The region is subdivided into $\lfloor (d-c)/(b-a) \rfloor + 2$ subdomains that can be treated with the algorithm in section 4. This convolution domain can be formed by concatenating Figure 5.1 and Figure 5.2, and can be computed by combining those two cases. The crosses show the evaluation points used to construct h^{mid} when $N = 11$.

$(d-a)/(b-a) > 2$. In total the domain is subdivided into $r+2$ pieces, $r-1$ of them being required for the first term in (5.2), and 3 of them for the second term.

Therefore, to compute $f_N * g_M$ we require $r+2$ calls to the algorithm described in section 4, each one requiring $\mathcal{O}((M+N)N)$ operations, and this leads to a computational cost that grows linearly with the ratio $(d-c)/(b-a)$. When working to a finite precision the total cost can be reduced, sometimes quite significantly, by compressing polynomial representations at several stages (see section 7).

The partitionings described here are continuous analogues of the *overlap-add method* used to evaluate discrete convolutions of vectors of length M and N, where $N \gg M$ [14, pp. 369–371]. In that discrete setting, the longest vector is divided into vectors of length M (analogous to our restrictions of g_N) and the resulting discrete convolutions summed up carefully to contribute to the final result. If M exactly divides into N (analogous to case (i)) the vector is exactly partitioned, otherwise there is a small convolution for the remaining $< M$ entries (analogous to case (ii)), which is computed by zero padding that vector to length M. The difference in our continuous setting is that we do not have the luxury of zero padding, and instead must restrict f_M.

6. Numerical results. Our algorithm is publicly available in the Chebfun command called `conv`, and the results below use that implementation [16]. The numerical experiments were performed on a single core of a 2011 1.8GHz Intel Core i7 MacBook Air with MATLAB 2014a. Execution times are approximate.

We first take two randomly generated Legendre series[2] of degree N, for 65 values of N that are logarithmically spaced[3] between 10 and 10^5 and compare the computational time for computing (4.2) using the quadrature approach (see section 2) and our algorithm. Figure 6.1 shows the computational times for the quadrature approach and our algorithm. The computational cost grows quadratically for our algorithm and cubically for the quadrature approach, matching their respective algorithmic complexities. For all the numerical experiments the norm of the difference between the two computed solutions was zero to 15 or 16 digits.

In section 5, we showed that the complexity of our algorithm depends on the ratio of the length of the intervals, whereas the quadrature based approach does not. In the regime of $(d-c) \gg (b-a)$ one would expect the quadrature approach to be more efficient, and to investigate this we compared the computational time required by the two algorithms for computing (1.2) for Legendre series of degree N on intervals

[2]Random vectors of Legendre coefficients are generated for (4.1) with the MATLAB command `randn` using the random number generator `mt19937ar` with the seed initialized to 1.

[3]The values of N we used are computed by the MATLAB code `ceil(logspace(1,5,65))`.

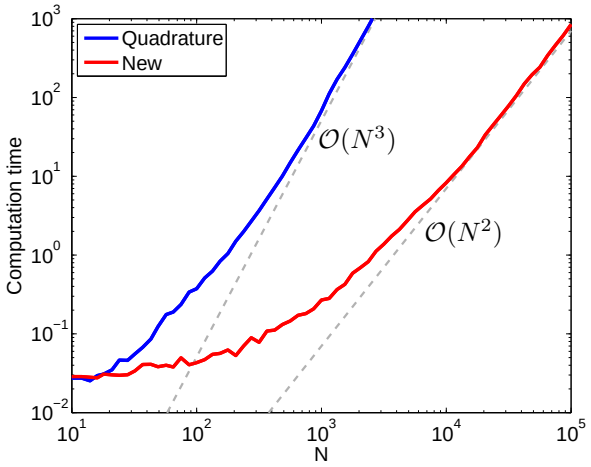

Fig. 6.1: Computational time for convolution of Legendre series on $[-1, 1]$ of degree N for the quadrature approach (blue) and our algorithm (red).

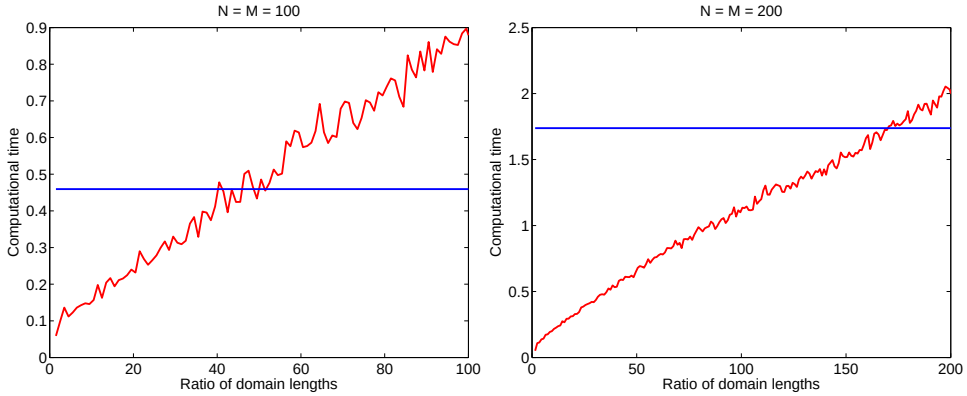

Fig. 6.2: Computation time for the convolution of Legendre series on $[-1, 1]$ and $[-r - 1/2, r + 1/2]$ for $r = 1, \ldots, N$ with $N = M = 100$ (left) and $N = M = 200$ (right). The execution time for the quadrature approach (blue) does not depend on the length of the intervals, whereas our algorithm (red) depends linearly on the ratio.

$[a, b] = [-1, 1]$ and $[c, d] = [-n - 1/2, n + 1/2]$ for $n = 1, \ldots, N$. Figure 6.2 shows the computational time for the two algorithms when $N = 100$ and $N = 200$. For $N = 100$ the quadrature approach is more efficient when $(d - c)/(b - a) > 50$ and for $N = 200$ when $(d - c)/(b - a) > 170$. By considering the complexities of the algorithms the ratio for which the two approaches are of similar efficiency should grow linearly with N.

7. Further details. A few extra details were required for the `conv` command in Chebfun [16], which we now describe.

Compression of polynomial representation. The algorithm we have described thus so far works in infinite precision arithmetic, and can be made faster in finite precision by adaptively pruning polynomial expansions to remove numerically insignificant trailing coefficients. We do this at several places in the implementation and almost all intermediary polynomials are compressed. This compression is most

beneficial when the original Legendre coefficients decay at least algebraically, such as when they are derived from a finite Legendre expansion of differentiable or analytic functions. In particular, we consider the computational timings reported in section 6 as worst case as the original Legendre coefficients are not derived from expansions of smooth functions.

Chebyshev series. The Chebfun software system represents a function defined on $[-1,1]$ by a Chebyshev series

$$p_N(x) = \sum_{n=0}^{N} \alpha_n^{cheb} T_n(x), \qquad x \in [-1,1],$$

where $T_n(x) = \cos(n \cos^{-1} x)$ is the Chebyshev polynomial of degree n [12]. This is extended to general bounded intervals $[a,b]$ by using the linear map $\psi_{[a,b]}$ in the same way as (4.9). Thus, the implementation in the `conv` command needs to compute the convolution of two Chebyshev series and return a piecewise polynomial represented by Chebyshev series. Therefore, the first and last part of our implementation transform between Chebyshev and Legendre series, which are rapidly computed with the `cheb2leg` and `leg2cheb` commands in Chebfun [7].

Currently, we are not aware of an algorithm for computing $f_M * g_N$ that works directly with Chebyshev series and has the same efficiency as we have shown in this article. Many of the relationships we exploited in section 3 become far more complicated with Chebyshev polynomials and involve the weight function $w(x) = (1-x^2)^{-1/2}$.

Piecewise smooth functions. Chebfun can also approximate piecewise smooth functions, by bundling together Chebyshev series defined on adjacent intervals. We can easily extend our algorithm to the convolution of piecewise smooth functions: Suppose $x_1 \le \cdots \le x_{K+1}$ such that f_M and g_N are polynomials of degree at most M and N on the subintervals $[x_k, x_{k+1}]$, $1 \le k \le K$, i.e.,

$$f_M = \sum_{k=1}^{K} f_M \chi_{[x_k,x_{k+1}]}, \qquad g_N = \sum_{j=1}^{K} g_N \chi_{[x_j,x_{j+1}]},$$

where $\chi_{[x_k,x_{k+1}]}$ is the indicator function for $[x_k, x_{k+1}]$. By the distributivity of '$*$', we have

$$(f_M * g_N) = \sum_{k=1}^{K} \sum_{j=1}^{K} \left(f_M \chi_{[x_k,x_{k+1}]} * g_N \chi_{[x_j,x_{j+1}]} \right),$$

where each term in the double sum is a convolution of two polynomials.

8. Appendix. The convolution algorithm we have described is applicable to Legendre series on general bounded intervals $[a,b]$ and $[c,d]$, with the canonical algorithm working when $[a,b] = [c,d] = [-1,1]$. Below we give the MATLAB code for computing the convolution of two Legendre series on $[-1,1]$. The inputs are Legendre coefficients for f_M and g_N and the outputs are the coefficients for h^{left} and h^{right}. A full implementation that is applicable to functions on general intervals and piecewise smooth functions is publicly available in the `conv` command in Chebfun [16].

```
function [gL, gR] = shortConv(a, b)
% Input:  Legendre coeffs of f_M and g_N.
```

```
% Output: Legendre coeffs of h^left and h^right.
    MN = length(a) + length(b);          % Maximum degree of result
    if ( length(b) > length(a) ), tmp = a; a = b; b = tmp; end
    a = [a ; zeros(MN - length(a), 1)];  % Pad to make length N
    % S represents multiplication by 1/z in spherical Bessel space:
    e = [[1 ; 1./(2*(1:(MN-1)).'+1)], ...
        [1 ; zeros(MN-1, 1)], -1./(2*(0:MN-1).'+1)];
    S = spdiags(e, -1:1, MN, MN);
    gL = rec(S, a, b, -1);               % Legendre coeffs of h^left
    S(1,1) = -1;                         % Modify S for right piece
    gR = -rec(S, a, b, 1);               % Legendre coeffs of h^right
end
function g = rec(S, a, b, sgn)           % Form B via rec st g=B(a)*b
    idx = 1:length(b);                   % Useful index
    scl = (-1).^(idx.'+1)./(2*idx.'-1);  % Scaling for upper-tri part
    %% First column of B:
    vNew = S*a; v = vNew;
    g = b(1)*vNew;
    bScl = b.*scl; bScl(1) = 0;
    g(1) = g(1) + vNew(idx).'*bScl;
    if ( length(b) == 1 ), return, end   % Scalar case is trivial!
    %% Second column of B:
    vNew = S*v + sgn*v; vOld = v; v = vNew; vNew(1) = 0;
    g = g + b(2)*vNew;
    bScl = -bScl*((2 - .5)/(2 - 1.5)); bScl(2) = 0;
    g(2) = g(2) + vNew(idx).'*bScl;
    %% Loop over remaining columns using recurrence:
    for k = 3:length(b)
        vNew = (2*k-3) * (S * v) + vOld; % Recurrence
        vNew(1:k-1) = 0;                 % Zero terms
        g = g + vNew*b(k);               % Append to g
        % Recurrence unstable for j<k. Correct for upper-tri part:
        bScl = -bScl*((k-.5)/(k-1.5)); bScl(k) = 0;
        g(k) = g(k) + vNew(idx).'*bScl;
        vOld = v; v = vNew;              % Seed new recurrence
    end
end
```

Acknowledgments. We are grateful to Rodrigo Platte for drawing our attention to the connection between the Fourier transform of Legendre polynomials and Bessel functions. We also thank Nick Trefethen for providing us with an interesting test problem [15, Chap. 6], which our new algorithm has sped up by a factor of more than 50. We have also benefited from discussions with members of the Chebfun team.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1965.

[2] B. K. Alpert and V. Rokhlin, *A fast algorithm for the evaluation of Legendre expansions*, SIAM J. Sci. Stat. Comp., 12 (1991), pp. 158–179.

[3] J. P. Boyd, *Finding the zeros of a univariate equation: proxy rootfinders, Chebyshev interpolation, and the companion matrix*, SIAM Review, 55 (2013), pp. 375–396.
[4] C. W. Clenshaw, *A note on the summation of Chebyshev series*, Math. Comp., 51 (1955), pp. 118–120.
[5] A. R. DiDonato, *Recurrence relations for the indefinite integrals of associated Legendre functions*, Math. Comp., 38 (1982), pp. 547–551.
[6] W. M. Gentleman, *Implementing Clenshaw–Curtis quadrature II: Computing the cosine transformation*, Commun. ACM, 15 (1972), pp. 343–346.
[7] N. Hale and A. Townsend, *A fast, simple, and stable Chebyshev–Legendre transform using an asymptotic formula*, to appear in SIAM J. Sci. Comp.
[8] R. V. Hogg, J. W. McKean, and A. T. Craig, *Introduction to Mathematical Statistics*, 6th edition, Prentice Hall, New Jersey, 2004.
[9] S. Jaroszewicz and M. Korzen, *Arithmetic operations on independent random variables: a numerical approach*, SIAM J. Sci. Comp., 34 (2012), A1241–A1265.
[10] Y. Katznelson, *An Introduction to Harmonic Analysis*, Dover, 1976.
[11] V. Madisetti and D. Williams, *Digital Signal Processing Handbook on CD-ROM*, CRC Press, 1999.
[12] J. C. Mason and D. C. Handscomb, Chebyshev Polynomials, CRC Press, 2002.
[13] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST Handbook of Mathematical Functions*, Cambridge University Press, 2010.
[14] S. Salivahanan and A. Vallavaraj, *Digital Signal Processing*, Tata McGraw-Hill Education, 2000.
[15] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013.
[16] L. N. Trefethen et al., Chebfun Version 5, The Chebfun Development Team, (2014), `http://www.chebfun.org/`.