Journal of Computational Physics 356 (2018) 98–114

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Multithreaded implicitly dealiased convolutions ☆

Malcolm Roberts [a], John C. Bowman [b,*]

[a] Computer Modelling Group Ltd, 3710 33 Street NW, Calgary, Alberta T2L 2M1, Canada
[b] Department of Mathematical and Statistical Sciences, University of Alberta, Edmonton, Alberta T6G 2G1, Canada

ARTICLE INFO

Article history:
Received 12 April 2017
Received in revised form 22 November 2017
Accepted 22 November 2017
Available online 2 December 2017

Keywords:
Convolution
Implicit dealiasing
Fast Fourier transform
Multithreading
Parallelization
Pseudospectral method

ABSTRACT

Implicit dealiasing is a method for computing in-place linear convolutions via fast Fourier transforms that decouples work memory from input data. It offers easier memory management and, for long one-dimensional input sequences, greater efficiency than conventional zero-padding. Furthermore, for convolutions of multidimensional data, the segregation of data and work buffers can be exploited to reduce memory usage and execution time significantly. This is accomplished by processing and discarding data as it is generated, allowing work memory to be reused, for greater data locality and performance. A multithreaded implementation of implicit dealiasing that accepts an arbitrary number of input and output vectors and a general multiplication operator is presented, along with an improved one-dimensional Hermitian convolution that avoids the loop dependency inherent in previous work. An alternate data format that can accommodate a Nyquist mode and enhance cache efficiency is also proposed.

1. Introduction

The convolution is an important operator in a wide variety of applications ranging from statistics, signal processing, image processing, and the numerical approximation of solutions to nonlinear partial differential equations. The convolution of two sequences $\{F_k\}_{k\in\mathbb{Z}}$ and $\{G_k\}_{k\in\mathbb{Z}}$ is $\sum_{p\in\mathbb{Z}} F_p G_{k-p}$. In practical applications, the inputs $\{F_k\}_{k=0}^{m-1}$ and $\{G_k\}_{k=0}^{m-1}$ are of finite length m, yielding a linear convolution with components $\sum_{p=0}^{k} F_p G_{k-p}$ for $k = 0, \ldots, m-1$. Computing such a convolution directly requires $\mathcal{O}(m^2)$ operations, and roundoff error is a significant problem for large m. It is therefore preferable to make use of the convolution theorem, harnessing the power of the fast Fourier transform (FFT) to map the convolution to a component-wise multiplication. This reduces the computational complexity of a convolution to $\mathcal{O}(m \log m)$ [6,8] while improving numerical accuracy [9].

Since the FFT considers the inputs to be periodic, the direct application of the convolution theorem results in a circular convolution, due to the indices being computed modulo m. Removing these extra aliases from the periodic convolution to produce a linear convolution is called *dealiasing*.

We give a brief overview of the dealiasing requirements for different types of convolutions in Section 2. The standard method for dealiasing FFT-based convolutions is to pad the inputs with a sufficient number of zero values such that the aliased contributions are all zero, as shown in Fig. 1. In Section 3, we generalize the method of implicit dealiasing [4] to handle an arbitrary number of input and output vectors, with a general spatial multiplication operator. This allows im-

☆ This work was supported by the Natural Sciences and Engineering Research Council of Canada.
* Corresponding author.
E-mail addresses: malcolm.i.w.roberts@gmail.com (M. Roberts), bowman@ualberta.ca (J.C. Bowman).

https://doi.org/10.1016/j.jcp.2017.11.026
0021-9991/

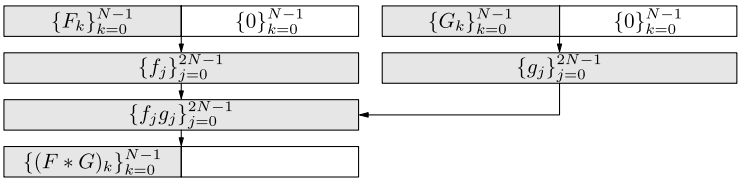

Fig. 1. Computing a 1D convolution via explicit zero padding.

plicit dealiasing to be efficiently applied to autocorrelations and pseudospectral simulations of nonlinear partial differential equations (e.g. in hydrodynamics and magnetohydrodynamics). We also discuss key technical improvements that allow implicit dealiasing to be fully multithreaded. For an efficient in-place implementation of the centered Hermitian convolution, it was necessary to unroll the outer loop partially so that interacting wavenumbers can be simultaneously processed. This loop unrolling offers another advantage: it removes the loop interdependence that prevented Function `conv` in [4] from being fully parallelized. For the construction of 2D and 3D convolutions, discussed in Section 4, the advantage of our new 1D convolution routines (relative to those in Ref. [4]) is the better pipelining afforded by loop interdependence, not their parallelizability, as the multithreading is now done at a higher level. The higher dimensional convolutions are decomposed into a sequence of lower-dimensional convolutions, each of which are run on a separate thread. We demonstrate that multithreaded implicit dealiasing in dimensions greater than one uses far less memory and is much faster than explicit dealiasing. The accomplishments of this work and future directions for research are summarized in Section 5. Implicitly dealiased convolution routines are publicly available in the open-source software library `FFTW++` [5], which is built on top of the widely used `FFTW` library [7].

2. Dealiasing requirements for convolutions

To compute the standard linear convolution $\sum_{p=0}^{k} F_p G_{k-p}$ for $k \in \{0, \ldots, m-1\}$, the data is padded with m zeroes for a total FFT length of $2m$. We refer to these inputs as *non-centered* and the paddings as $1/2$ *padding*. If the input data is multidimensional with size $m_1 \times \ldots \times m_d$, then the data must be zero padded to $2m_1 \times \ldots \times 2m_d$, increasing the buffer size by a factor of 2^d.

For pseudospectral simulations, it is convenient to shift the zero wavenumber in the transformed data to the middle of the array. In this case, the inputs are $\{F_k\}_{k=-m+1}^{m-1}$ and $\{G_k\}_{k=-m+1}^{m-1}$, which we refer to as *centered*, and their convolution has components $\sum_{p=k-m+1}^{m-1} F_p G_{k-p}$ for $k = -m+1, \ldots, m-1$. Convolutions on centered inputs require less padding than on non-centered inputs: data of length $2m-1$ needs to be padded only to length $3m-2$ (normally extended to $3m$); this is called $2/3$ *padding* [11]. Explicit zero padding increases the d-dimensional buffer size in this case by a factor of $(3/2)^d$.

A binary convolution can be generalized to an n-ary operator $*(F_1, \ldots, F_n)_k = \sum_{p_1,\ldots,p_n} F_{p_1} \cdots F_{p_n} \delta_{p_1+\ldots+p_n,k}$, where δ is the Kronecker delta. For non-centered inputs, an n-ary convolution could be computed as a sequence of binary convolutions using $1/2$ padding. However, for centered inputs with both negative and positive frequencies, each binary convolution would have to be padded further to eliminate all aliased interactions [12]. As a result, n-ary convolutions benefit greatly from implicit dealiasing [4].

We consider a generalized convolution operation that takes A inputs and produces B outputs, where the multiplication performed in the transformed space can be an arbitrary component-wise operation. In order to make use of $1/2$ padding or $2/3$ padding (for noncentered or centered inputs, respectively), the multiplication operator must be quadratic; if the multiplication operator is of higher degree, one must extend the padding to remove undesired aliases. To compute a convolution with A inputs and B outputs using the convolution theorem, one performs A backward FFTs to transform the inputs, applies the appropriate multiplication operation on the transformed data, and then performs B forward FFTs to produce the final outputs, for a total of $A+B$ FFTs.

The choice of multiplication operator determines the type of convolution. Let $\{f_j\}$ be the inverse Fourier transform of $\{F_k\}$. An autoconvolution can be computed with just two transforms using $A = B = 1$ and the operation $f_j \to f_j^2$, while an autocorrelation would use $f_j \to f_j \bar{f}_j$, where $\bar{f}_j$ denotes the complex conjugate of f_j. For the standard binary convolution, there are two inputs and one output, and the multiplication operation is $(f_j, g_j) \to f_j g_j$.

The nonlinear advective term of the 2D incompressible Navier–Stokes vorticity equation can be computed with the operation $(u_x, u_y, \partial\omega/\partial x, \partial\omega/\partial y) \to (u_x \partial\omega/\partial x + u_y \partial\omega/\partial y)$, where $\boldsymbol{u} = (u_x, u_y)$ is the 2D velocity and $\omega = \hat{\boldsymbol{z}} \cdot \nabla \times \boldsymbol{u}$ is the z-component of the vorticity; this requires a total of five FFTs ($A = 4$ and $B = 1$). As shown in Appendix A, it is possible to reduce the FFT count for this case to four, with $A = B = 2$. Similarly, in three dimensions, Basdevant [2] showed that the number of FFT calls can be reduced from nine to eight, with $A = 3$ and $B = 5$. For incompressible 3D magnetohydrodynamic (MHD) flows the operation is $(\boldsymbol{u}, \boldsymbol{\omega}, \boldsymbol{B}, \boldsymbol{j}) \to (\boldsymbol{u} \times \boldsymbol{\omega} + \boldsymbol{j} \times \boldsymbol{B}, \boldsymbol{u} \times \boldsymbol{B})$, where $\boldsymbol{u}$ is the velocity, $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$ is the vorticity, $\boldsymbol{B}$ is the magnetic field, and $\boldsymbol{j}$ is the current density ($A = 12$, $B = 6$) [13]. However, in Appendix A we show that Basdevant's technique can be used to reduce the number of calls to 17 ($A = 6$, $B = 11$). For the Navier–Stokes and MHD equations, the operation is quadratic and the convolution is binary ($n = 2$), with a padding ratio of $2/3$ (since the Fourier modes are