Implicitly Dealiased Convolutions for DNS: Preliminary MPI results

John Bowman and Malcolm Roberts

University of Alberta, Aix-Marseille University

Euromech 542, 2013-01-17

Properties

Convolutions are binary operations on the set of L^2 functions.
For $f, g : \mathbb{R}^d \to \mathbb{F}$, with $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$,

- Commutative: $f * g = g * f$.
- Associative: $(f * g) * h = f * (g * h)$.
- Identity element: Dirac delta:

$$f * \delta = \delta * f = f.$$

- Transitive: $(f + g) * h = f * g + g * h$.
- Easily extendable to functions on $\mathbb{R}^d \to \mathbb{F}$.

Convolution Theorem

Let

$$F(k) = \int dx\, e^{2\pi i\, kx} f(x) \doteq \mathcal{F}[f](k)$$

and $G(k) = \mathcal{F}[g](k)$. Then

$$\begin{aligned}
\mathcal{F}[f * g](k) &= \int dx\, e^{2\pi i\, kx} \int dy\, f(y) g(x-y) \\
&= \int dy \int dx'\, f(y) e^{2\pi i\, k(y+x')} g(x') \\
&= \int dy\, e^{2\pi i\, ky} f(y) \int dx'\, e^{2\pi i\, kx'} g(x') \\
&= \mathcal{F}[f](k) \times \mathcal{F}[g](k).
\end{aligned}$$

Discrete convolutions

For $f, g \in \ell^2$,

$$(f * g)_n = \sum_{\ell=0}^{n} f_\ell g_{n-\ell}$$
$$= \sum_{\ell_1=0}^{n} \sum_{\ell_2=0}^{n} f_{\ell_1} g_{\ell_2} \delta_{\ell_1+\ell_2,n}$$

This requires $\mathcal{O}(N^2)$ operations for data of length N.

It is better to compute using a fast Fourier transform using $\mathcal{O}(N \log N)$ operations:

$$f * g = \mathcal{F}^{-1}(\mathcal{F}[f] \times \mathcal{F}[g]).$$

Non-centered data

Data is often *non-centered*, i.e.

$$\{f_n\}_{n=0}^{N-1} : \{0, N-1\} \to \mathbb{F}.$$

Binary convolutions can be easily extended to higher-order:

$$\begin{aligned} *(f, g, h)_n &= \sum_{a,b,c=0}^{N-1} f_a g_b h_d \delta_{a+b+c,n} \\ &= f * (g * h). \end{aligned}$$

Most applications use this type of data.

Discrete Convolution Theorem

Let $\zeta_N = e^{2\pi i/N}$ denote the N^{th} root of unity.
The inverse Fourier transform of $\{F_k\}_{k=0}^{N-1}$ is

$$f_j = \mathcal{F}^{-1}[F_k]_j = \sum_{k=0}^{N-1} \zeta_N^{jk} F_k$$

The orthogonality of the transform relies on the fact that

$$\sum_{j=0}^{N-1} \zeta_N^{\ell j} = \begin{cases} N & \text{if } \ell = sN \text{ for } s \in \mathbb{Z}, \\ \frac{1-\zeta_N^{\ell N}}{1-\zeta_N^{\ell}} = 0 & \text{otherwise.} \end{cases}$$

Aliasing Errors

For non-centered data, we get

$$\sum_{j=0}^{N-1} \zeta_N^{-kj} F_k G_k = \sum_{j=0}^{N-1} \zeta_N^{-kj} \left(\sum_{p=0}^{N-1} \zeta_N^{kp} f_p \right) \left(\sum_{q=0}^{N-1} \zeta_N^{kq} g_q \right)$$

$$= \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} f_p g_q \sum_{j=0}^{N-1} \zeta_N^{k(p+q-j)}$$

$$= N \sum_{s\in\mathbb{Z}} \sum_{p=0}^{N-1} f_p g_{q-j+sN}$$

The terms with $s \neq 0$ are called *aliasing errors*.
The product is a cyclic convolution, with indices $\bmod N$.

Explicit zero-padding

The convolution can be *dealiased* by extending the input data with a bunch of zeros:

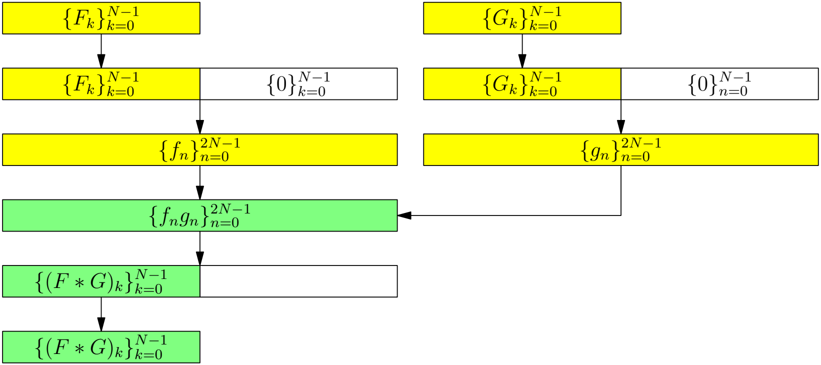
Pad from length N to length $2N$:

$$\{\widetilde{F}_k\}_{n=0}^{2N-1} = (F_0, F_1, \ldots, F_{N-2}, F_{N-1}, \underbrace{0, \ldots, 0}_{N})$$

$$\begin{aligned}(\widetilde{F} *_{2N} \widetilde{G})_k &= \sum_{\ell=0}^{2N-1} \widetilde{F}_{\ell \pmod{2N}} \widetilde{G}_{(k-\ell) \pmod{2N}} \\ &= \sum_{\ell=0}^{N-1} F_\ell \widetilde{G}_{(k-\ell) \pmod{2N}} \\ &= \sum_{\ell=0}^{k} F_\ell G_{k-\ell}.\end{aligned}$$

Multidimensional convolutions are padded in each direction.

Explicit zero-padding



Implicit zero-padding

Implicit padding involves using a separate work array to compute the DFT:

$$f_x = \sum_{k=0}^{2N-1} \zeta_{2N}^{xk} F_k, \quad F_k = 0 \text{ if } k \geq N$$

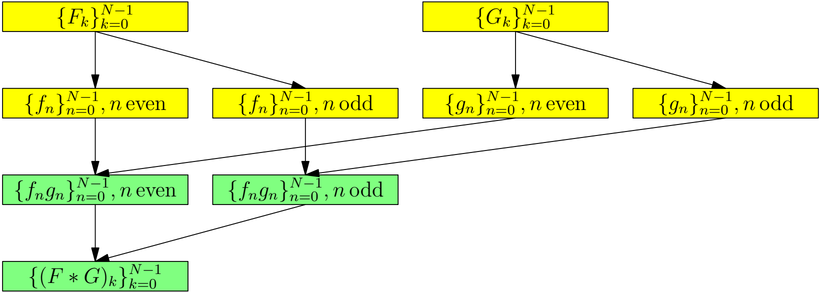
is attained by computing

$$f_{2x} = \sum_{k=0}^{N-1} \zeta_N^{xk} F_k$$

and

$$f_{2x+1} = \sum_{k=0}^{N-1} \zeta_N^{xk} (\zeta_{2N}^{x} F_k).$$

Implicit zero-padding



Convolutions on $\mathbb{R}^d$

For 2D convolutions, one performs FFTs in each direction:

$$\mathcal{F}_y^{-1}\left\{\mathcal{F}_x^{-1}\left(\mathcal{F}_x\left[\mathcal{F}_y\{f\}\right] \times \mathcal{F}_x\left[\mathcal{F}_y\{g\}\right]\right)\right\}.$$

However, an x-convolution is just

$$\mathcal{F}_x^{-1}\left(\mathcal{F}_x[f] \times \mathcal{F}_x[g]\right).$$

Which is to say that we perform the operation

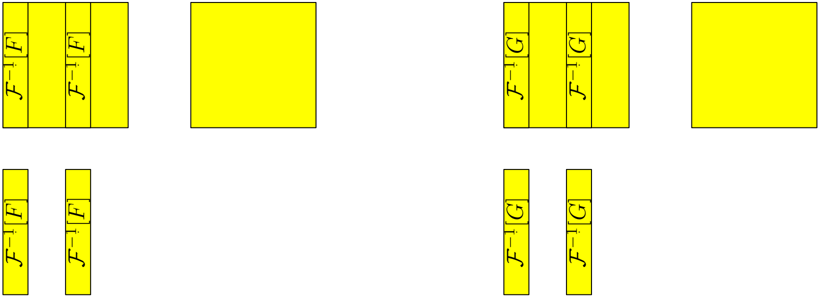
$$\mathcal{F}_y \to x\text{-convolution} \to \mathcal{F}_y^{-1}.$$

This allows us to re-use work arrays with subconvolutions when using implicit padding.

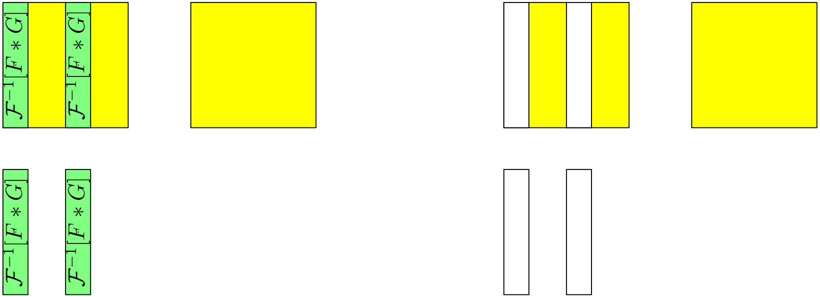
Implicit Zero-padding

Implicit Zero-padding

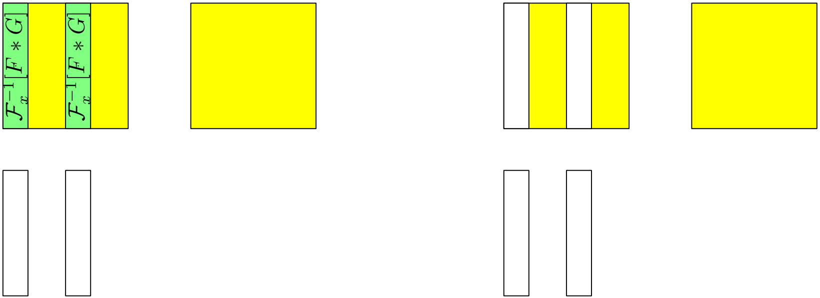
Implicit Zero-padding



Implicit Zero-padding



Implicit Zero-padding



Implicit Zero-padding

Implicit Zero-padding

Comparison of techniques: non-centered

- Implicit padding uses

$$\left(\frac{1}{2}\right)^{d-1}$$

  the memory of explicit padding.

- The algorithm is about twice as fast for $d > 1$.

- Implicit and explicit padding exhibit similar numerical error.

Non-centered data

Consider the PDE

$$\frac{\partial u}{\partial t} = u \times u. \tag{1}$$

The Fourier transform of (1) is

$$\frac{\partial U}{\partial t} = U * U,$$

where $U = \mathcal{F}[u]$, and, for periodic domains,

$$U = \{U_k\}_{k=-N+1}^{N-1},$$

and $U_{-k} = U_k^*$, where * denotes complex conjugation: the data exhibits Hermitian symmetry.

Centered convolutions

The convolution of such input F and G is

$$(F * G)_k = \sum_{\ell=k-m+1}^{m-1} F_\ell G_{k-\ell}$$

Inputs are zero-padded using a "2/3" rule: the input data, $\{F_k\}_{k=-N+1}^{N-1}$, is padded from length $2N-1$ to length $3N$.

Higher-order convolutions (e.g. ternary) are not associative:

$$*(F, G, H) \neq F * (G * H),$$

and must be computed all-at-once and further padded.

Comparison of techniques: centered

- Implicit padding uses

$$\left(\frac{2}{3}\right)^{d-1}$$

  the memory of explicit padding.

- The algorithm is about twice as fast for $d > 1$.
- Implicit and explicit padding exhibit similar numerical error.
- The advantage of implicit padding increases for higher-order convolutions.

MPI

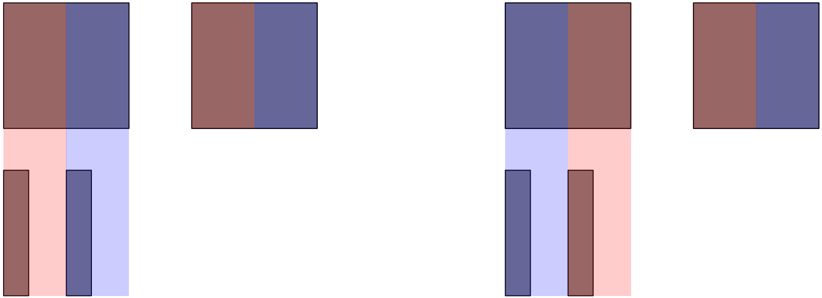
Implicit padding skips transforms on zero-data while maintaining a good data structure. This also reduces the communication cost using MPI.

- Memory savings translate into communication savings.
- Communication done via FFTW's MPI transpsoe
- 3D convolutions can be done with either 1D or 2D decompositions.
- Parallelization is hybrid OpenMP/MPI.

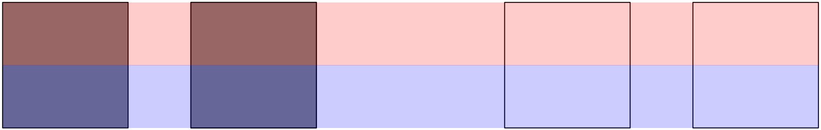
MPI

MPI

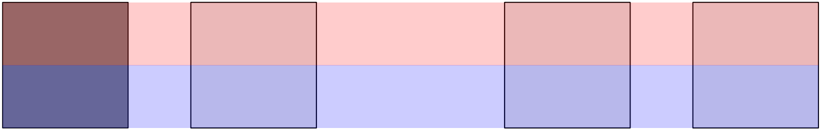
MPI



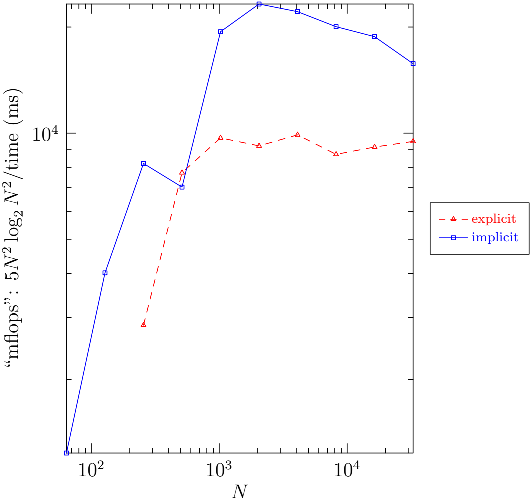
MPI



MPI

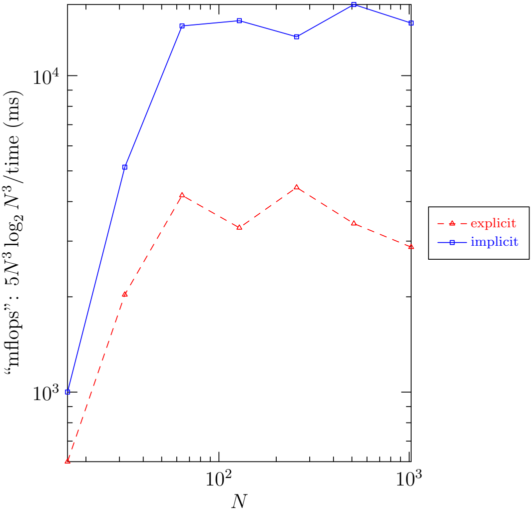


MPI: speed



2D convolution on 192 cores.

MPI: speed



3D convolution on 192 cores.

Conclusions

- Implicitly dealiased convolutions are faster and use less memory.
- A multi-threaded implementation is available at `fftwpp.sourceforge.net`, using `FFTW` for Fourier transforms.
- Written in `C++`, wrappers for `C`, Python, and `Fortran`.
- A hybrid OpenMP/MPI implementation is soon to be released.

Future Work

- Finalize tests and release MPI version.
- Convolutions on real data.
- Special cases (e.g. self-convolution).
- Impliment in as part of a MPI pseudospectral solver.
- Can implicit padding be used for phase-shift dealiasing?
- Can we extend implicit padding to other basis functions?