The Fastest Convolution in the West

Malcolm Roberts and John C. Bowman

Aix-Marseille University, University of Alberta

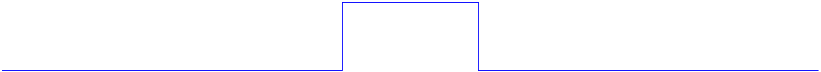
CEMRACS, 2012-08-14

Convolutions

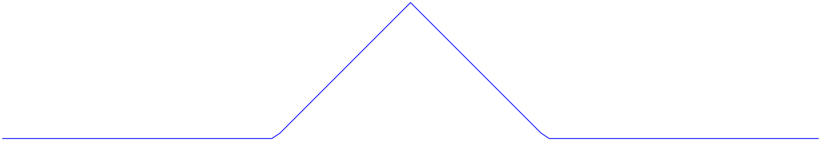
The convolution of the functions F and G is

$$(F * G)(t) = \int_{-\infty}^{\infty} F(\tau)G(t-\tau)\,d\tau.$$

For example, if $F = G = \chi_{(-1,1)}(t)$



Then $F * G$ is:



Convolutions

- Out-of-focus images are a convolution.
- Image filtering.
- Digital signal processing.
- Correlation analysis.
- The Lucas–Lehmer primality test uses fast convolutions.
- Pseudospectral simulations of nonlinear PDEs.

Convolutions

The convolution of $F = \{F_k\}_{k\in\mathbb{Z}}$ and $G = \{G_k\}_{k\in\mathbb{Z}}$ is denoted $F * G$, with

$$(F * G)_k = \sum_{\ell, m \in\mathbb{Z}} F_\ell G_m \delta_{k,\ell+m} = \sum_{\ell \in\mathbb{Z}} F_\ell G_{k-\ell}$$

Properties:

- Commutativity: $F * G = G * F$
- Associativity: $*(F, G, H) = (F * G) * H = F * (G * H)$, where

$$*(F, G, H)_k = \sum_{\ell_1,\ell_2,\ell_3\in\mathbb{Z}} F_{\ell_1} G_{\ell_2} H_{\ell_3} \delta_{k,\ell_1+\ell_2+\ell_3}$$

- Identify element: $F * \delta = \delta * F = F$

Application: Correlation Analysis

- The cross-correlation of F and G is $F \star G$, with

$$(F \star G)_k = \sum_{\ell} F_\ell^* G_{k+\ell}.$$

- This can be computed as the convolution of F_k^* with G_{-k}.
- Cross-correlation is useful in signal processing and data analysis.
- In this case, input data is $\{F_k\}_{k=0}^{N-1}$, or *non-centered*.

Non-centered data

- Input data: $\{F_k\}_{k=0}^{N-1}$ and $\{G_k\}_{k=0}^{N-1}$.
- This produces non-centered convolutions:

$$(F * G)_k = \sum_{\ell=0}^{k} F_\ell G_{k-\ell}, \quad k = 0, \ldots, N-1$$

- For non-centered data, $*(F, G, H) = F * (G * H) = (F * G) * H$.

Application: Pseudospectral simulations

- The incompressible 2D Navier–Stokes vorticity equation

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\,\omega = \nu \nabla^2 \omega$$

is Fourier-transformed into

$$\frac{\partial \omega_k}{\partial t} = \sum_{p+q=k} \frac{\epsilon_{kpq}}{q^2} \omega_p^* \omega_q^* - \nu k^2 \omega_k, \quad \epsilon_{kpq} = (\hat{z} \cdot p \times q)\delta_{k+p+q}$$

- The nonlinearity becomes a convolution:

$$(F * G)_k = \sum_{k_1, k_2} F_{k_1} G_{k_2} \, \delta_{k, k_1, k_2}.$$

Application: Pseudospectral simulations

- Input data $\{F_k\}_{k=-N+1}^{N-1}$ is *centered*.
- It is also *Hermitian-symmetric* $F_{-k} = F_k^*$.
- Hermitian symmetry $\iff \mathcal{F}^{-1}[F] \in \mathbb{R}$

Centered data

- Input data: $\{F_k\}_{k=-N+1}^{N-1}$ and $\{G_k\}_{k=-N+1}^{N-1}$.

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

- Considering Hermitian-symmetric data ($F_{-k} = F_k^*$), we compute data for $k \geq 0$, so

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

Centered data

Theorem

For centered data, $*(F, G, H) \neq F * (G * H) \neq (F * G) * H$.

Proof.

Let $N = 2$.

$*(F_a, G_b, H_c)_1$			$(F_a * (G_b * H_c)_\ell)_1$			
a	b	c	a	ℓ	b	c
1	0	0	1	0	0	0
0	1	0	0	1	1	0
0	0	1	0	1	0	1
1	1	-1	1	0	1	-1
1	-1	1	1	0	-1	1
-1	1	1	N/A			

$\square$

FFT-based convolutions

- The convolution sum involves $\mathcal{O}(N^2)$ terms. Using FFTs, we can compute a convolution in $\mathcal{O}(N \log N)$ operations.
- The inverse discrete Fourier transform (DFT) of $\{F_k\}_{k=0}^{N-1}$ is

$$f_n \doteq \mathcal{F}^{-1}[F] = \sum_{k=0}^{N-1} \zeta_N^{nk} F_k$$

- $\zeta_N = e^{\frac{2\pi i}{N}}$ is the N^{th} root of unity. $\zeta_{aN}^{a} = \zeta_N$, $\zeta_N^N = 1$.
- For $\{F_k\}_{k\in\mathbb{Z}}$, $\{G_k\}_{k\in\mathbb{Z}}$,

$$\mathcal{F}[F * G] = \mathcal{F}[F] \times \mathcal{F}[G].$$

FFT-based convolutions

- The discrete Fourier transform treats arrays as periodic.
- A naive application of the convolution theorem produces a cyclic convolution:

$$\{F *_N G\}_k \doteq \sum_{\kappa=0}^{N-1} F_{\kappa \bmod N} G_{(k-\kappa) \bmod N},$$

- These extra terms are called *aliases*.

Dealiasing techniques

I compare three dealiasing techniques:

- Phase-shift dealiasing
- Explicit zero-padding
- Implicit zero-padding

Phase-shift dealiasing

The Δ-shifted Fourier transform,

$$\mathcal{F}_\Delta^{-1}[F]_j \doteq \sum_{k=0}^{m-1} \zeta_N^{(j+\Delta)k} F_k,$$

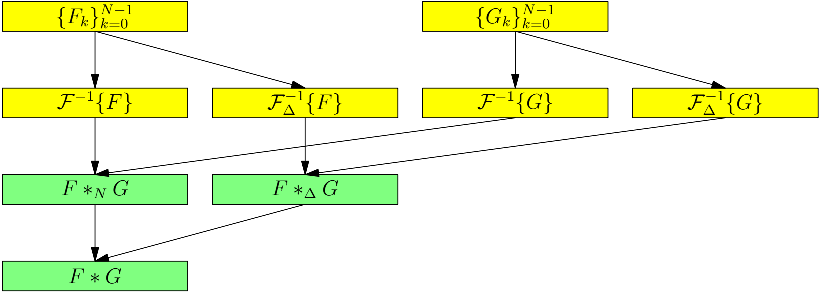
produces a convolution with an aliasing error of opposite sign for $\Delta = 1/2$:

$$\begin{aligned}\{F *_{\Delta} G\}_k &= \mathcal{F}_\Delta\left[\mathcal{F}_\Delta^{-1}[F] \times \mathcal{F}_\Delta^{-1}[F]\right] \\ &= \sum_{\kappa=0}^{k} F_\kappa G_{k-\kappa} - \sum_{\kappa=k+1}^{m-1} F_\kappa G_{k-\kappa+m}.\end{aligned}$$

One recovers the linear convolution by computing

$$F * G = \frac{1}{2}[(F *_{N} G) + (F *_{\Delta} G)].$$

Phase-shift dealiasing



Explicit zero-padding

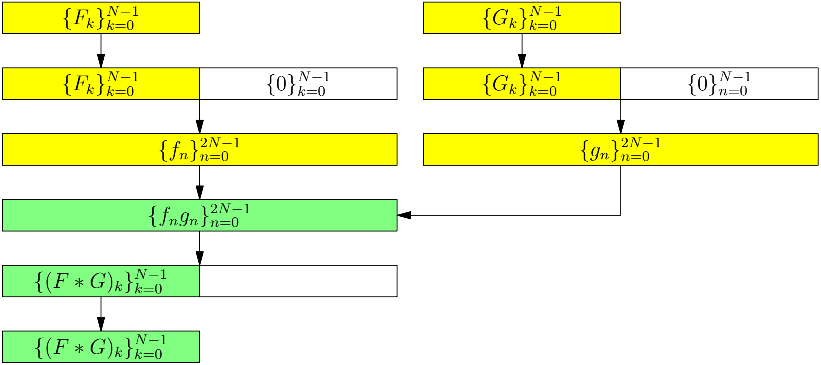
Another option is to append zero-data to the input array.
For non-centered data, pad from length N to length $2N$:

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

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

Centered data is padded from length $2N-1$ to length $3N$.

Explicit zero-padding



Implicit Zero-padding

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

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

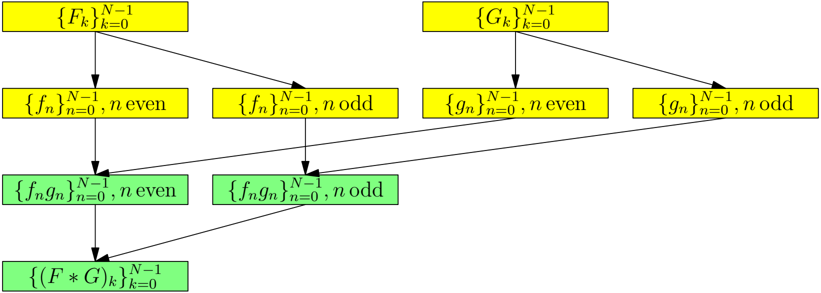
is attained by computing

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

and

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

Implicit zero-padding



Comparison of 1D methods: centered data

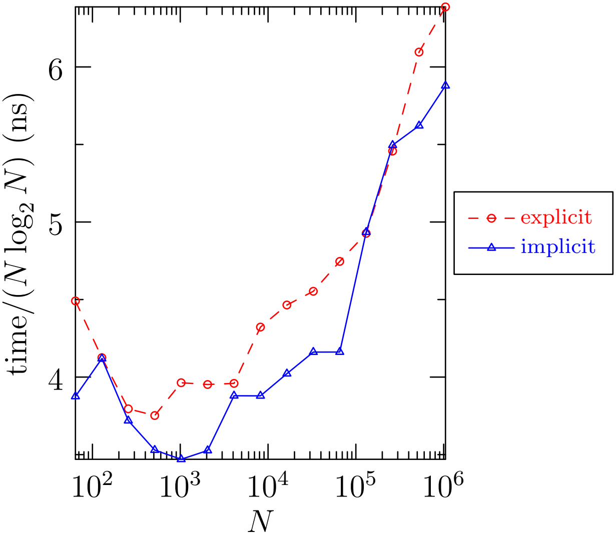
For non-centered data:

	Phase-shift dealiasing	Explicit padding	Implicit padding
Memory	$4N$	$4N$	$4N$
Complexity	$6KN\log N$	$6KN\log N$	$6KN\log N$

For centered Hermitian data:

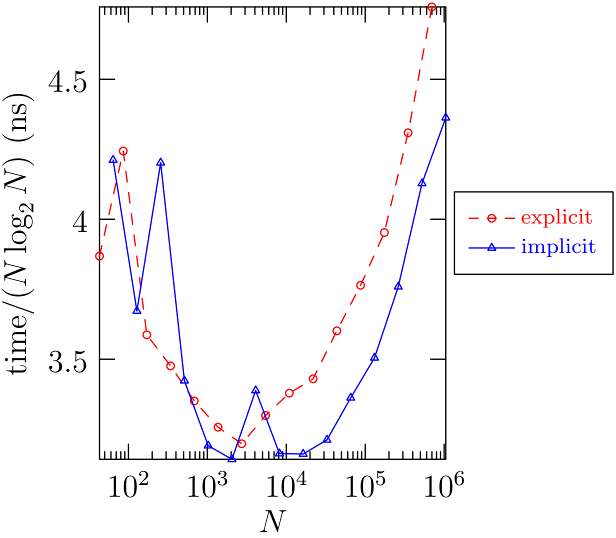
	Phase-shift dealiasing	Explicit padding	Implicit padding
Memory	$4N$	$3N$	$3N$
Complexity	$6KN\log N$	$\frac{9}{2}KN\log N$	$\frac{9}{2}KN\log N$

Comparison of zero-padding methods



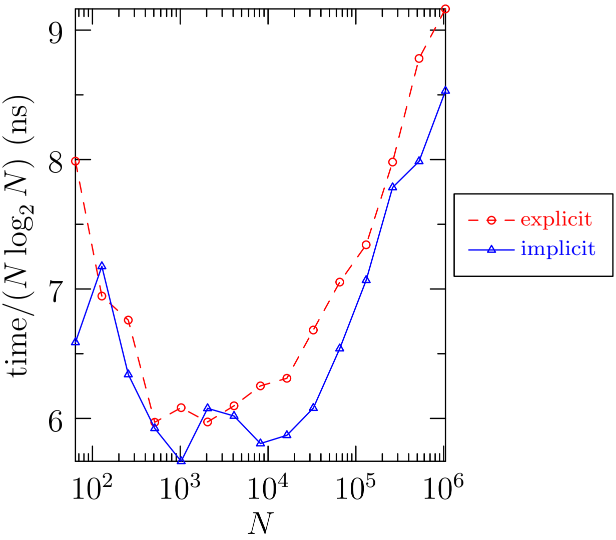
Complex non-centered 1D convolution.

Comparison of zero-padding methods



Hermitian-symmetric centered 1D convolution.

Comparison of zero-padding methods



Hermitian-symmetric centered 1D ternary convolution.

Phase-shift dealiasing: multiple dimensions

A d-dimensional convolution requires computing 2^d cyclic convolutions with different shifts.

For 3D pseudospectral simulations, one instead computes

$$F *_N G$$

and

$$F *_{\boldsymbol{\Delta}} G$$

with $\boldsymbol{\Delta} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. This removes singly-aliased terms.

Doubly and triply aliased terms are removed by setting terms to zero with $k \geq \frac{2\sqrt{2}}{3}N \approx 0.94N$.

Explicit Zero-padding: multiple dimensions

Multi-dimensional convolutions need to be padded in each dimension.
Non-centered convolutions are padded from N^d to

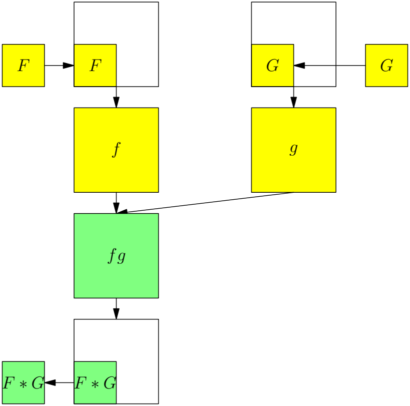
$$(2N)^d.$$

Centered convolutions are padded from $(2N-1)^d$ to

$$(3N)^d.$$

Some transforms are performed on arrays or zeroes; these can be skipped, and the transform is referred to as *pruned*.

Explicit Zero-padding: multiple dimensions



Implicit Zero-padding: multiple dimensions

The 2D FFT-based convolution algorithm is:

$$\mathcal{F}_y^{-1} \to \mathcal{F}_x^{-1} \to (\text{multiply}) \to \mathcal{F}_x \to \mathcal{F}_y$$

Note that

$$\mathcal{F}_x^{-1} \to (\text{multiply}) \to \mathcal{F}_x$$

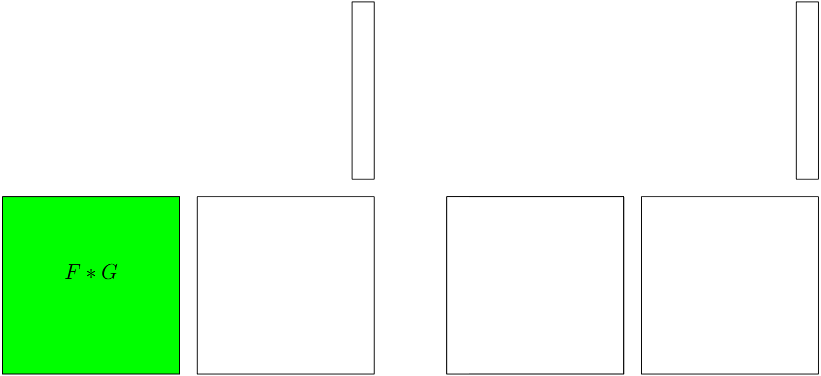
is just a convolution in the x-direction.

So the 2D convolution algorithm can be written

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

Since the implicitly dealiased convolution uses non-contiguous memory, we can re-use work arrays for sub-convolutions.

Implicit Zero-padding: multiple dimensions



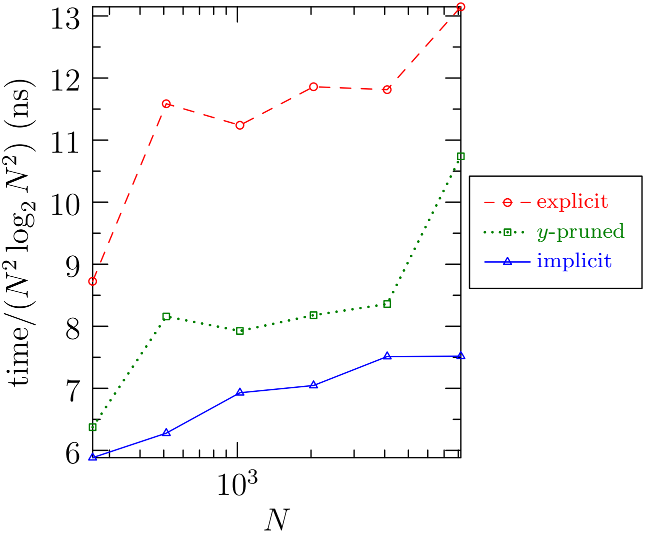
Comparison for Centered Convolutions

Method	Complexity	Memory Footprint
Explicit without Pruning	$3 \cdot 2^d d\, KN^d \log N$	$2^{d+1} N^d$
Explicit with Pruning	$6\left(2^d - 1\right) KN^d \log N$	$2^{d+1} N^d$
Implicit	$6\left(2^d - 1\right) KN^d \log N$	$4\, N^d$

Comparison for Centered Convolutions

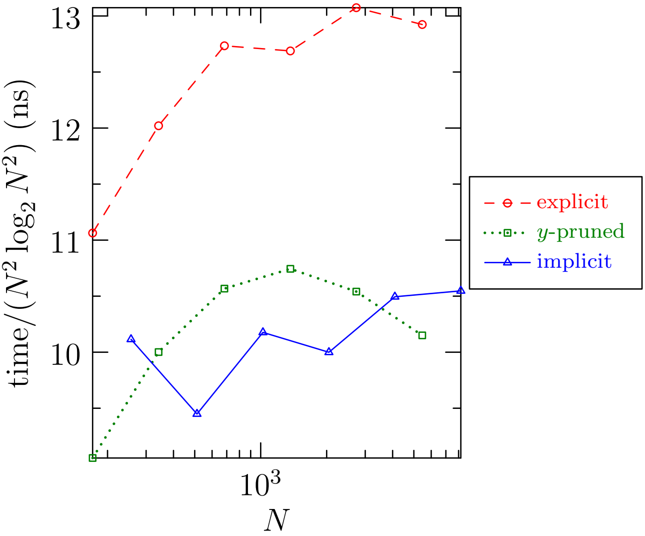
Method	Complexity	Memory Footprint
Phase-Shift Dealiasing	$3 \cdot 2^{2d-1} dKN^d \log N$	$2^{2d} N^d$
Partial Phase-shift	$3 \cdot 2^d dKN^d \log N$	$2^{d+1} N^d$
Explicit	$\frac{3^{d+1}}{2} d\, KN^d \log N$	$3^d N^d$
Explicit with Pruning	$\frac{9}{2}\left(3^d - 2^d\right)\, KN^d \log N$	$3^d N^d$
Implicit	$\frac{9}{2}\left(3^d - 2^d\right)\, KN^d \log N$	$3 \cdot 2^{d-1} N^d$

Performance: multiple dimensions



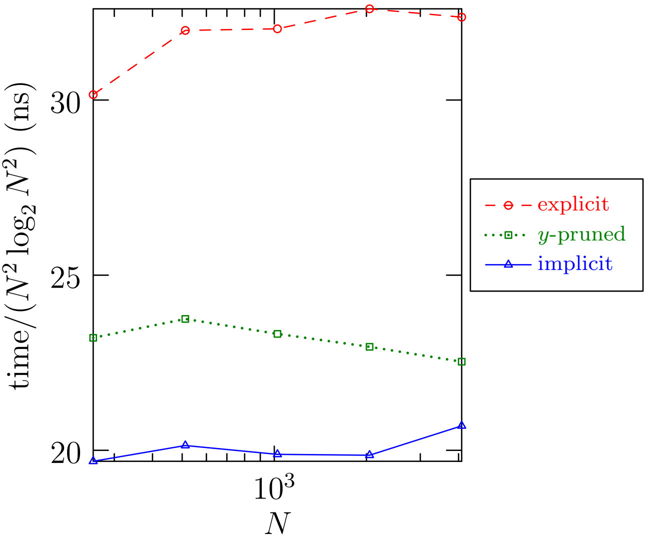
Complex non-centered 2D convolution.

Performance: multiple dimensions



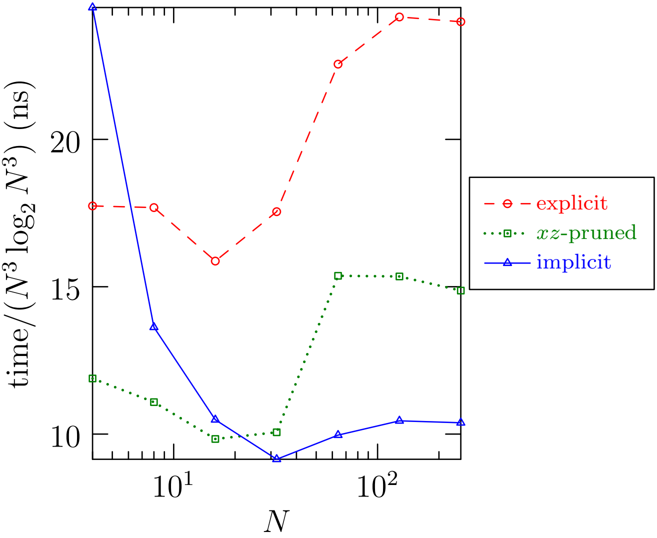
Hermitian-symmetric centered 2D convolution.

Performance: multiple dimensions



Hermitian-symmetric centered 2D ternary convolution.

Performance: multiple dimensions



Complex non-centered 3D convolutions.

Multi-threaded convolutions

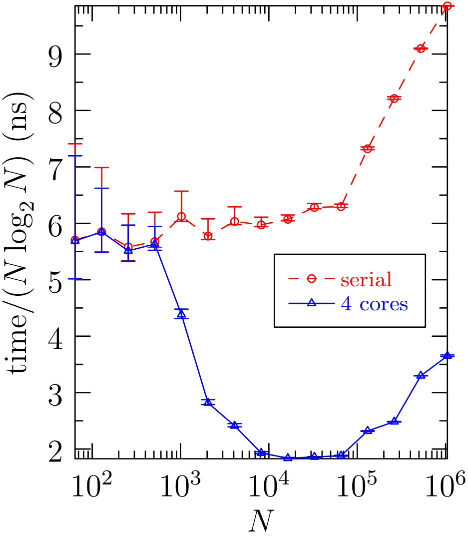
Implicit dealiasing has been implemented with multiple threads.
Each sub-convolution requires its own work array.
With P processors, the memory increase is of the order

$$PN^{d-1}$$

for d-dimensional convolutions.

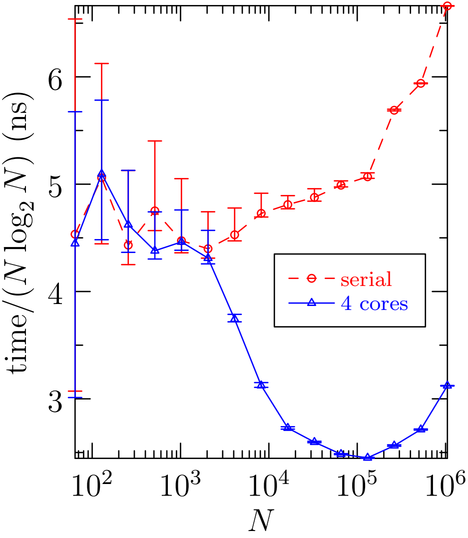
Implicit Zero-padding: multiple threads

Implicit multi-threading performance



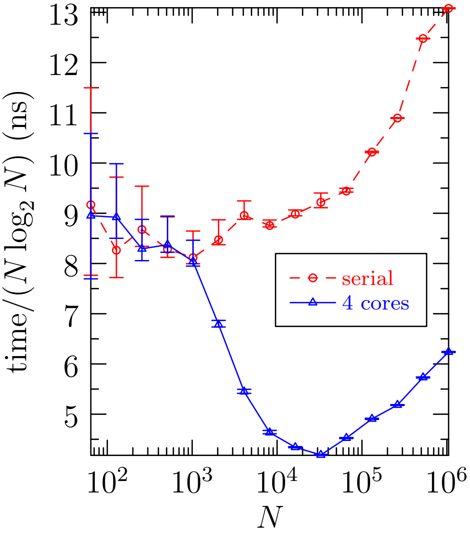
Non-centered 1D convolution.

Implicit multi-threading performance



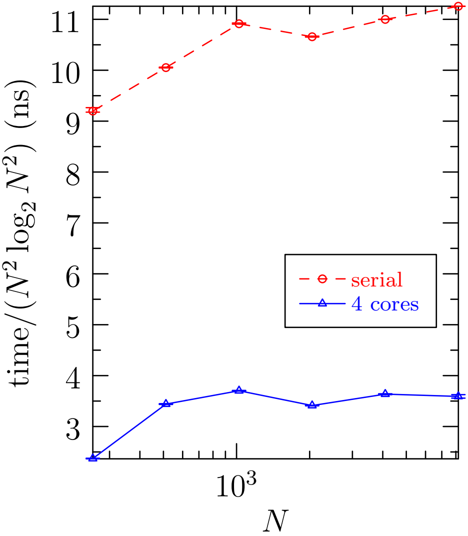
Centered 1D convolution.

Implicit multi-threading performance



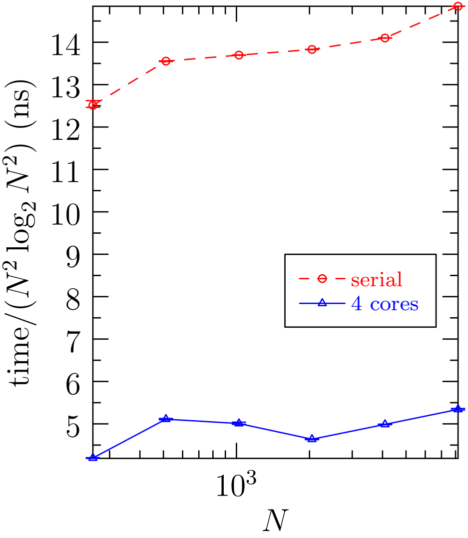
Centered 1D ternary convolution.

Implicit multi-threading performance



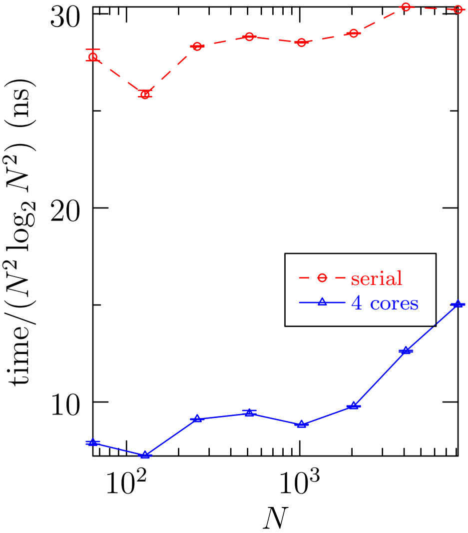
Non-centered 2D convolution.

Implicit multi-threading performance



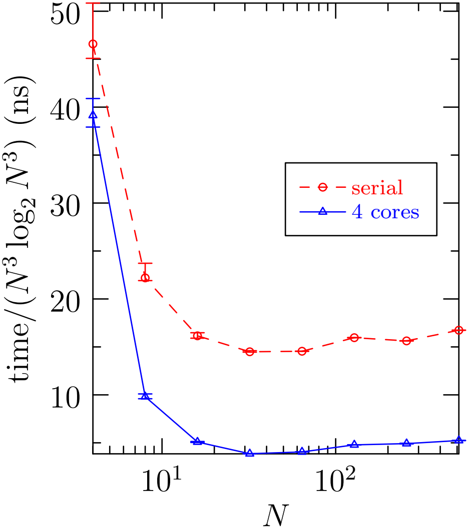
Centered 2D convolution.

Implicit multi-threading performance



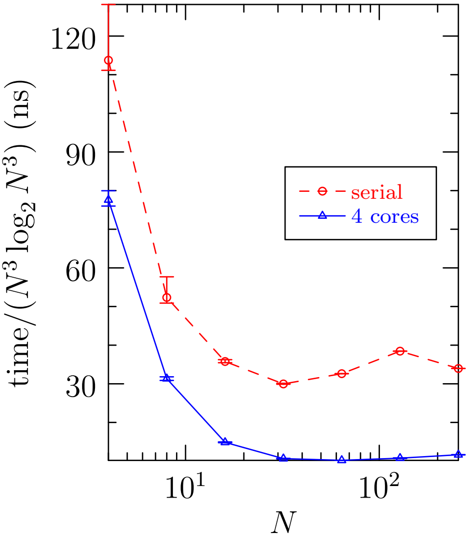
Centered ternary 2D convolution.

Implicit multi-threading performance



Non-centered 3D convolution.

Implicit multi-threading performance

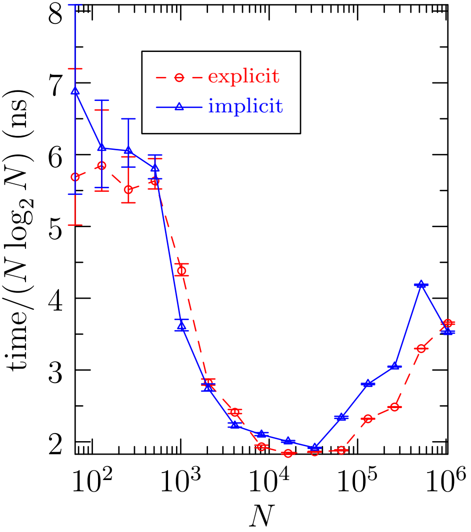


Centered 3D convolution.

Implicit multi-threading performance

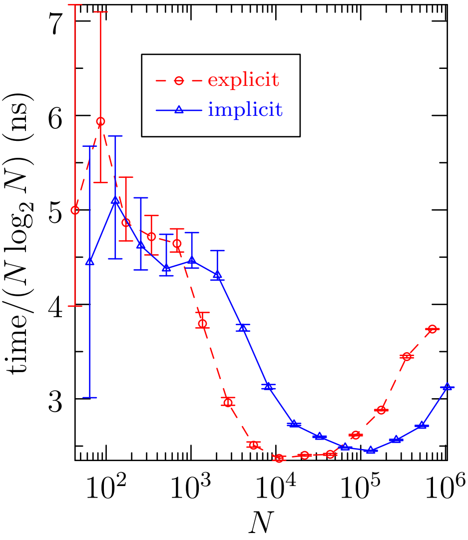
- One-dimensional convolutions on four cores are about 2 times as fast as on one core.
- Two-dimensional convolutions on four cores are about 3 times as fast.
- Three-dimensional convolutions on four cores are about 3.5 times as fast.

Multiple threads: explicit vs. implicit



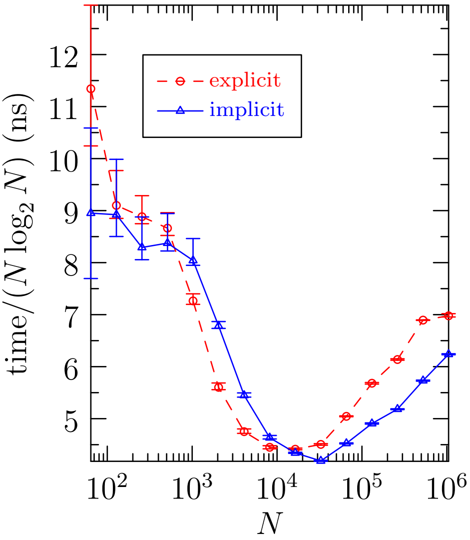
Non-centered 1D convolution.

Multiple threads: explicit vs. implicit



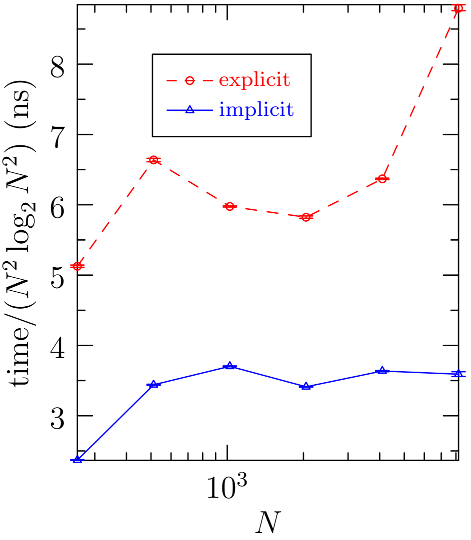
Centered 1D convolution.

Multiple threads: explicit vs. implicit



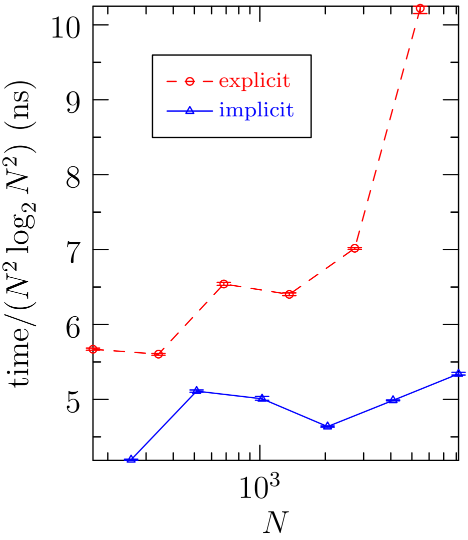
Centered ternary 1D convolution.

Multiple threads: explicit vs. implicit



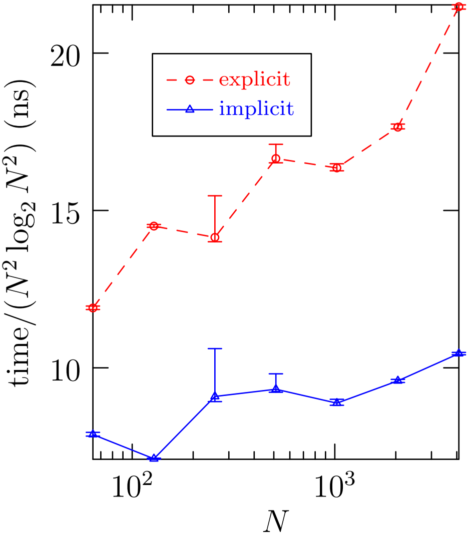
Non-centered 2D convolution.

Multiple threads: explicit vs. implicit



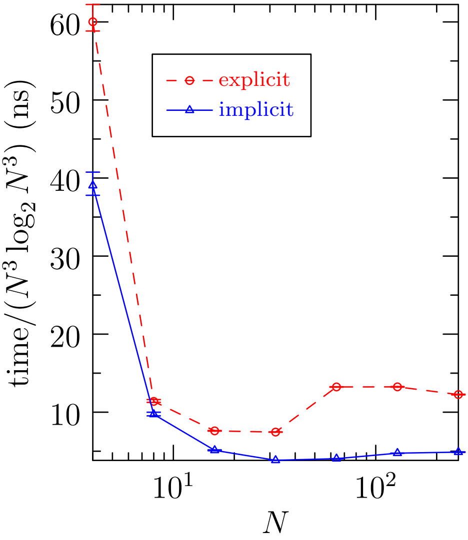
Centered 2D convolution.

Multiple threads: explicit vs. implicit



Centered ternary 2D convolution.

Multiple threads: explicit vs. implicit



Non-centered 3D convolution.

Summary of Results

- Implicit methods require much less work memory than is required by explicit methods .
- The implicit method had a speedup of up to 3.5 on four cores, while the explicit method sped-up of up to a factor of 3.
- The implicit method is around twice as fast as the explicit method for multidimensional convolutions.

Usage example

Computing the nonlinear source of the 2D incompressible Navier–Stokes equations in a vorticity formulation, which appears in Fourier space as

$$\sum_{p} \frac{p_x k_y - p_y k_x}{|k-p|^2} \omega_p \omega_{k-p},$$

is performed as follows:

$$\texttt{conv2}(ik_x\omega, ik_y\omega, ik_y\omega/k^2, -ik_x\omega/k^2).$$

One also has the option of passing work arrays to `conv2`, `which can then be used elsewhere.`

Conclusion

- Implicitly zero-padding multi-dimensional convolutions is faster and requires less memory than explicit routines.
- The algorithm has been successfully implemented on a shared-memory architecture with only a small increase in work memory.
- Convolution algorithms are available for complex non-centered data and centered Hermitian-symmetric data in 1D, 2D, and 3D.
- Ternary convolution algorithms are available for centered Hermitian-symmetric in 1D and 2D.

Future work

- A distributed-memory implementation based on openMPI.
- Improve multi-threaded parallelization.
- Convolutions on real data.
- Correlation routines.
- Auto-convolution/correlation routines.

Resources

`FFTW++`:
`http://fftwpp.sourceforge.net`

Asymptote:
`http://asymptote.sourceforge.net`

Malcolm Roberts:
`http://www.math.ualberta.ca/~mroberts`