The Fastest Convolution in the West

John Bowman and Malcolm Roberts

University of Alberta

June 16, 2010

`www.math.ualberta.ca/~mroberts`

Outline

- Convolution
  - Definition
  - Applications
- Fast Convolutions
  - The Convolution Theorem
- Aliasing Errors
  - Zero-padding
  - Phase-shift dealiasing
  - Implicit Padding
- Centered Hermitian Convolutions
- Ternary Convolutions

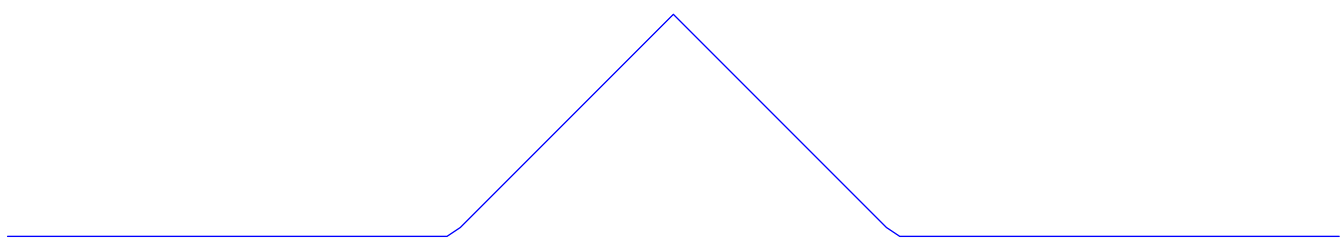
Convolutions

- 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:



Applications

- Out-of-focus images are a convolution:
  - the actual image is convolved with the aperture opening.
- Image filtering:
  - Sobel edge detection is a convolution of the image with a gradient stencil.
- Digital signal processing:
  - e.g. for low- and high-pass filters.
- Correlation analysis.
- The Lucas–Lehmer primality test uses fast convolutions.
  - Useful for testing Mersenne primes.
- Pseudospectral simulations of fluids:
  - $(u \cdot \nabla)u$ is a convolution in Fourier space.

Discrete Convolutions

- Applications use a *discrete linear convolution*:

$$(f * g)_n = \sum_{m=0}^{n} f_m g_{n-m}.$$

- Calculating $\{(f * g)_n\}_{n=0}^{N-1}$ takes $\mathcal{O}(N^2)$ operations.

- The convolution theorem states that convolutions are multiplications when Fourier-transformed:

$$\mathcal{F}[f * g] = \mathcal{F}[f]\,\mathcal{F}[g]$$

where $\{\mathcal{F}[f]\}_k = \sum_{n=0}^{N-1} e^{\frac{-2\pi i}{N}kn} f_n$ is the Fourier transform of f.

- A fast Fourier transform (FFT) of length N requires $KN\log_2 N$ multiplications [Gauss 1866], [Cooley & Tukey 1965].

- Convolving using FFTs requires $3KN\log_2 N$ operations.

Cyclic and Linear Convolutions

- Fourier transforms map periodic data to periodic data.
- Thus, $\mathcal{F}^{-1}[\mathcal{F}[f]\,\mathcal{F}[g]]$ is a *discrete cyclic convolution*,

$$(f *_N g)_n \doteq \sum_{m=0}^{N-1} f_{m(\mathrm{mod}\,N)}\, g_{(n-m)(\mathrm{mod}\,N)}.$$

- The difference between linear and cyclic convolutions,

$$(f *_N g)_n = \sum_{m=0}^{n} f_m g_{n-m} + \sum_{m=n+1}^{N-1} f_m g_{n-m+N},$$

is called the *aliasing error*.

Dealiasing via Explicit Zero-Padding

- The cyclic and linear convolutions are equal if we pad f with zeros:

$$\{\widetilde{f}_n\}_{n=0}^{2N-1} = (f_0, f_1, \ldots, f_{N-2}, f_{N-1}, \underbrace{0, \ldots, 0}_{N})$$

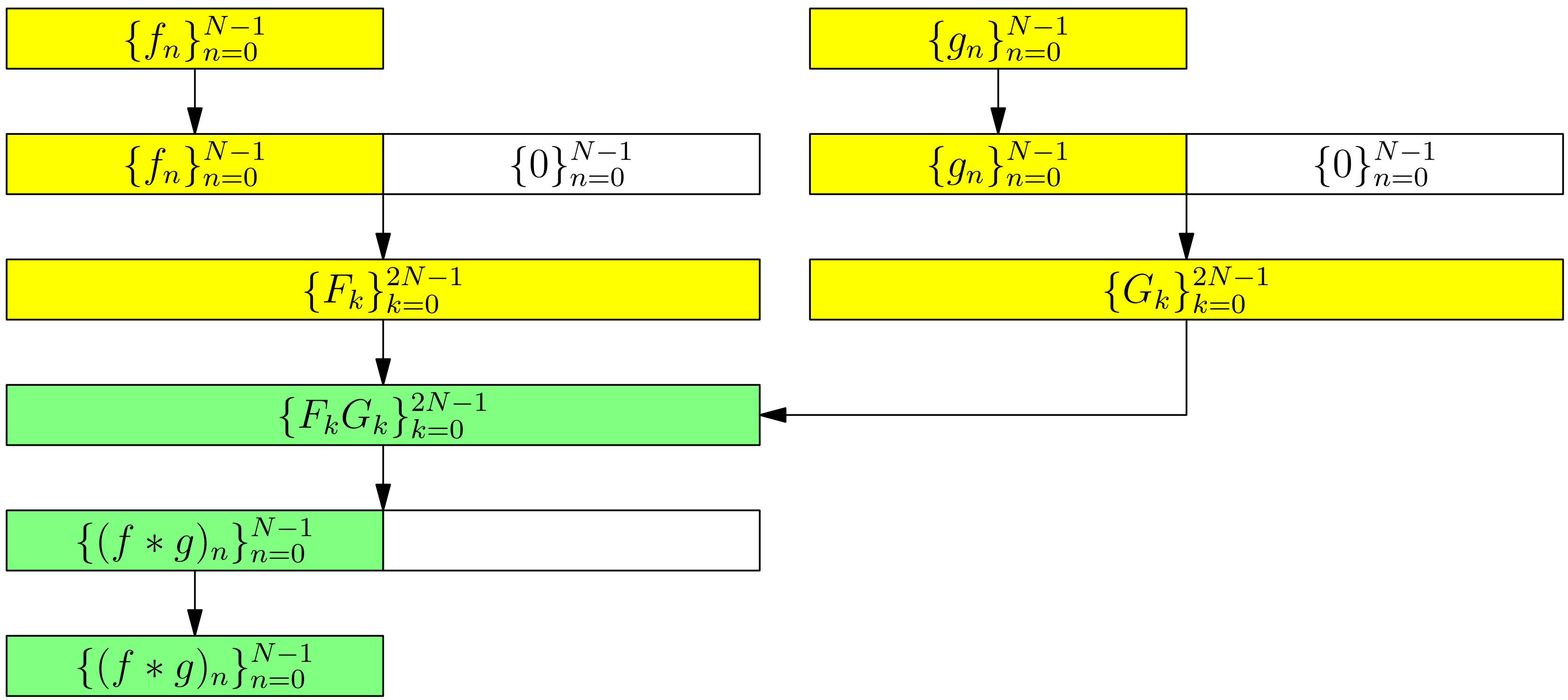
- Then,

$$(\widetilde{f} *_{2N} \widetilde{g})_n = \sum_{m=0}^{2N-1} \widetilde{f}_{m \pmod{2N}} \widetilde{g}_{(n-m) \pmod{2N}},$$

$$= \sum_{m=0}^{N-1} f_m \widetilde{g}_{(n-m) \pmod{2N}},$$

$$= \sum_{m=0}^{n} f_m g_{n-m}.$$

Dealiasing via Explicit Zero-Padding



- Convolving these padded arrays takes $6KN\log_2 2N$ operations,
- and twice the memory of a circular convolution.
- CPU speed and memory size have increased much faster than memory bandwidth; this is the *von-Neumann bottleneck*.

Phase-shift Dealiasing

- The shifted Fourier transform [Patterson Jr & Orszag 1971] is

$$F^{\Delta} \doteq \{\mathcal{F}^{\Delta}[f]\}_k = \sum_{n=0}^{N-1} e^{-\frac{2\pi i}{N}(k+\Delta)n} f_n.$$

- Then, setting $\Delta = 1/2$, one has

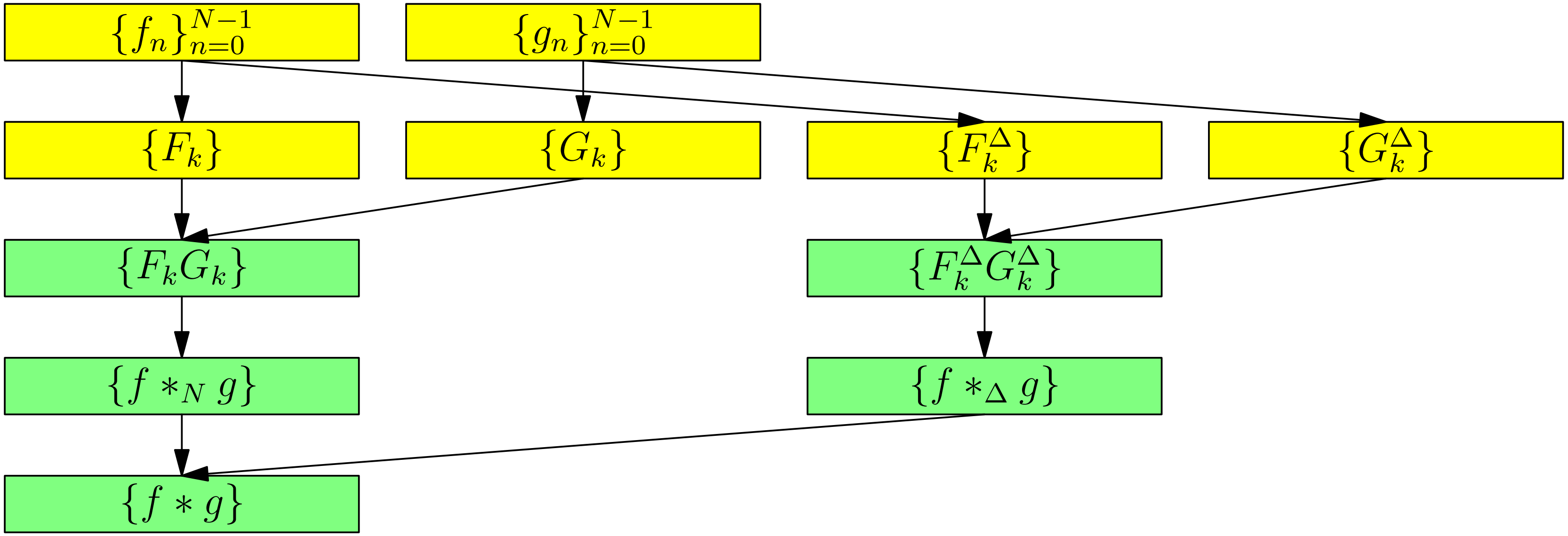
$$f *_{\Delta} g \doteq \mathcal{F}^{\Delta^{-1}}\left(F^{\Delta} G^{\Delta}\right) = \sum_{m=0}^{n} f_m g_{n-m} - \sum_{m=n+1}^{N-1} f_m g_{n-m+N},$$

which has a dealiasing error with opposite sign.

- We recover $f * g$ from two periodic convolutions:

$$f * g = \frac{1}{2}\left(f *_N g + f *_{\Delta} g\right).$$

Phase-shift Dealiasing



- We don't need to copy data to a larger buffer first.
- Convolving these padded arrays takes $6KN \log_2 N$ operations,
- The memory footprint is the same as explicit padding.
- Explicit padding is better if we need to add fewer than N zeros.

Implicit Padding

- Suppose that we want to take a Fourier transform of

$$\{f_n\}_{n=0}^{2N-1}, \text{ with } f_n = 0 \text{ if } n \geq N.$$

- The discrete Fourier transform is a sum:

$$\mathcal{F}(f)_k = \sum_{n=0}^{2N-1} e^{-\frac{2\pi i}{2N}kn} f_n.$$

- Since $f_n = 0$ if $n \geq N$, this is just

$$\mathcal{F}(f)_k = \sum_{n=0}^{N-1} e^{-\frac{2\pi i}{2N}kn} f_n.$$

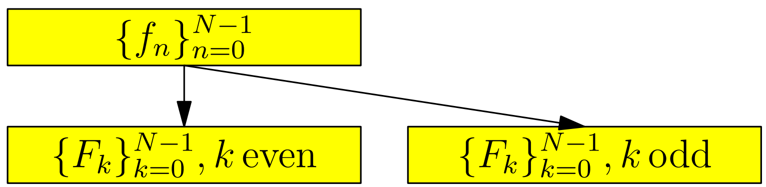
- This is not a Fourier transform: the FFT algorithm does not apply.

Implicit Padding

- However, if we calculate even and odd terms separately, we get

$$F_{2k} = \sum_{n=0}^{N-1} e^{-\frac{2\pi i}{N}kn} f_n, \qquad F_{2k+1} = \sum_{n=0}^{N-1} e^{-\frac{2\pi i}{N}kn} f_n e^{-\frac{2\pi i}{N}n},$$

which *are* Fourier transforms.



- The inverse is the sum of two Fourier transforms:

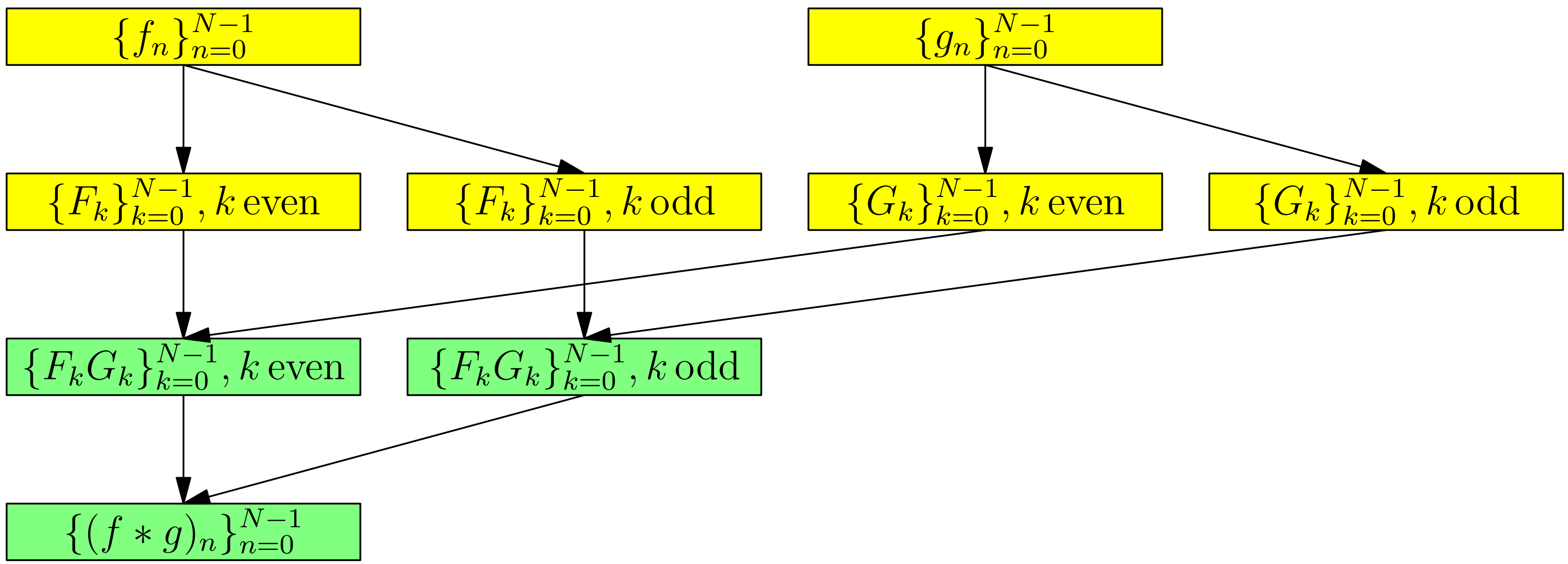
$$f_n = \frac{1}{N}\left(\sum_{k=0}^{N-1} e^{\frac{2\pi i}{N}kn} F_{2k} + e^{\frac{\pi i}{N}n} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N}kn} F_{2k+1}\right).$$

- Since Fourier-transformed data is of length $2N$, there are no memory savings.

Implicit Padding

- There is one advantage:

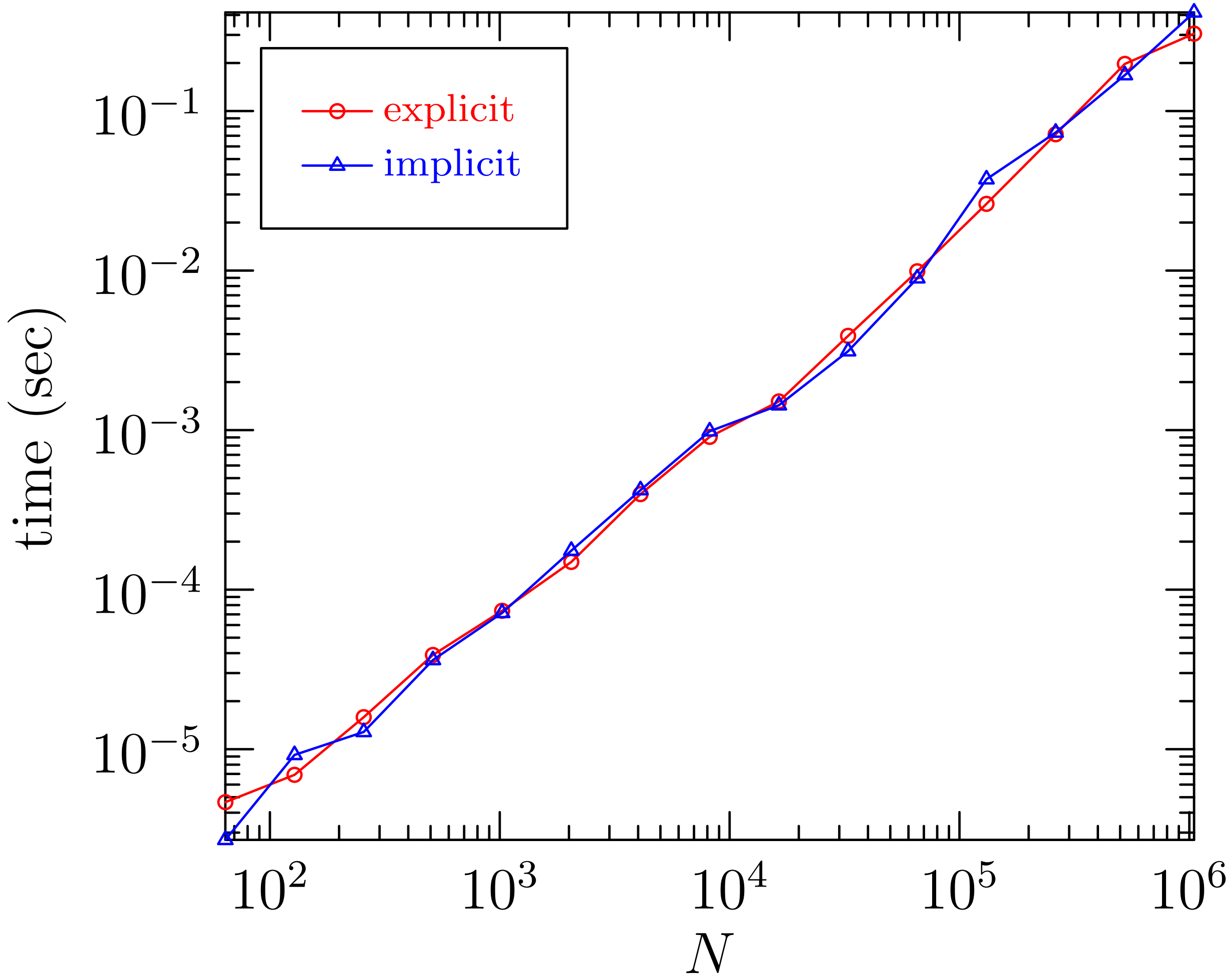
  the work buffer is separate from the data buffer.



- The computational complexity is $6KN\log_2 N/2$.
- By swapping arrays, we can use out-of-place transforms.
- The numerical error is similar to explicit padding.

Implicit Padding: speed

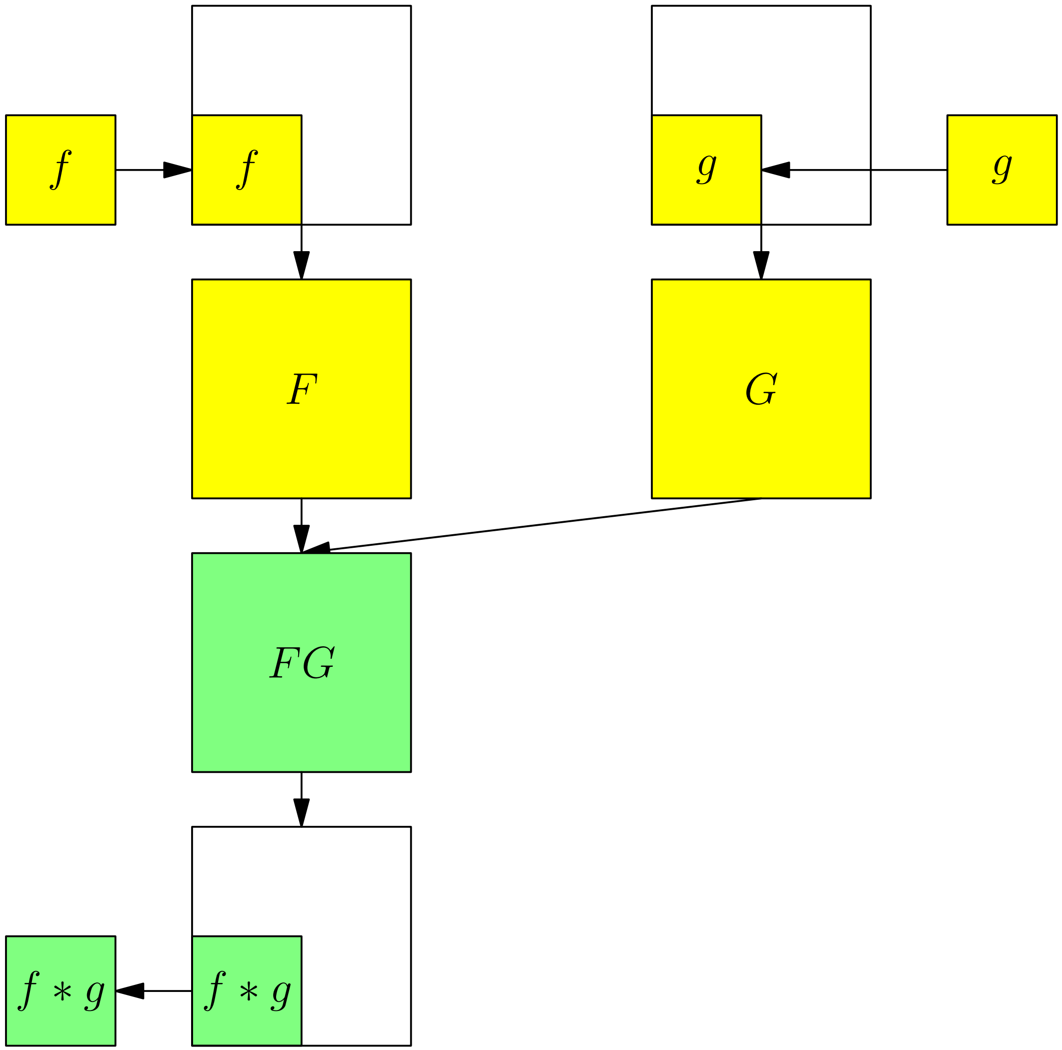
- The algorithms are comparable in speed:



- Ours is much more complicated.

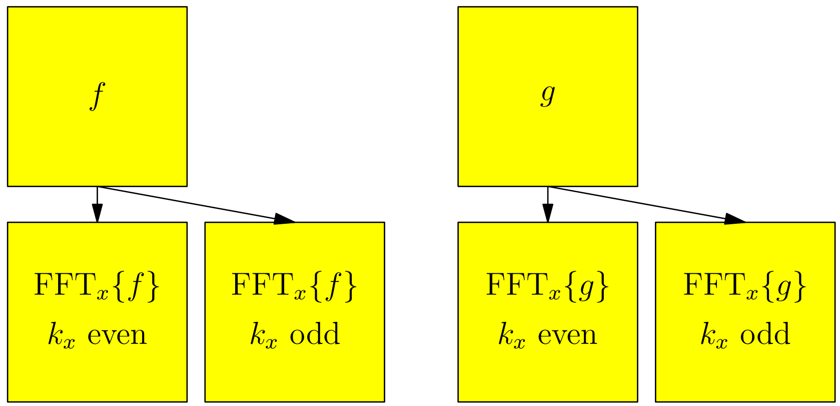
Convolutions in Higher Dimensions

- An explicitly padded convolution in 2 dimensions requires $12N$ padded FFTs, and 4 times the memory of a cyclic convolution.

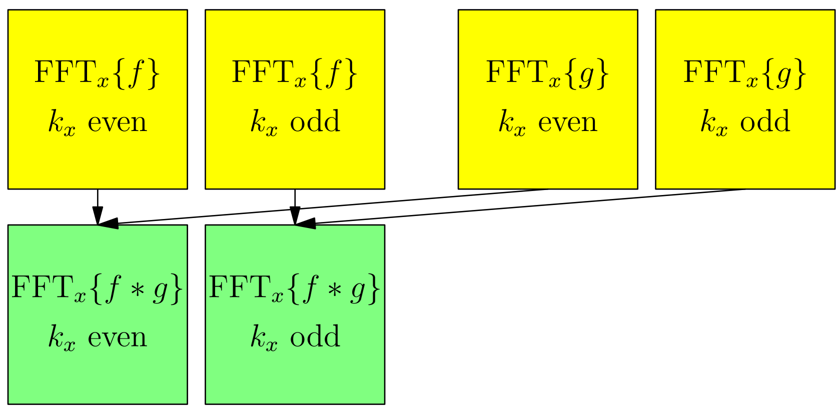


Implicit Convolutions in Higher Dimensions

- Implicitly padded 2-dimensional convolutions are done by first doing implicitly padded FFTs in the x direction:

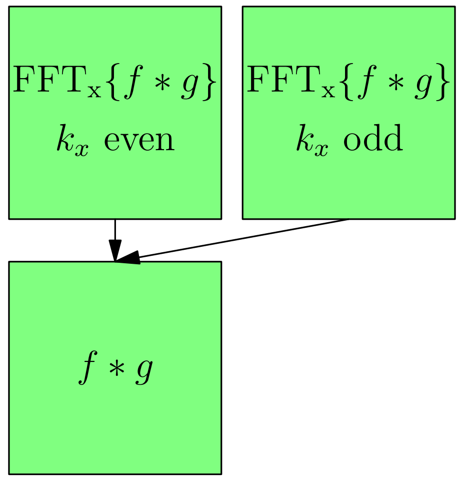


- And then $2N$ one-dimensional convolutions in the y-direction:



Implicit Convolutions in Higher Dimensions

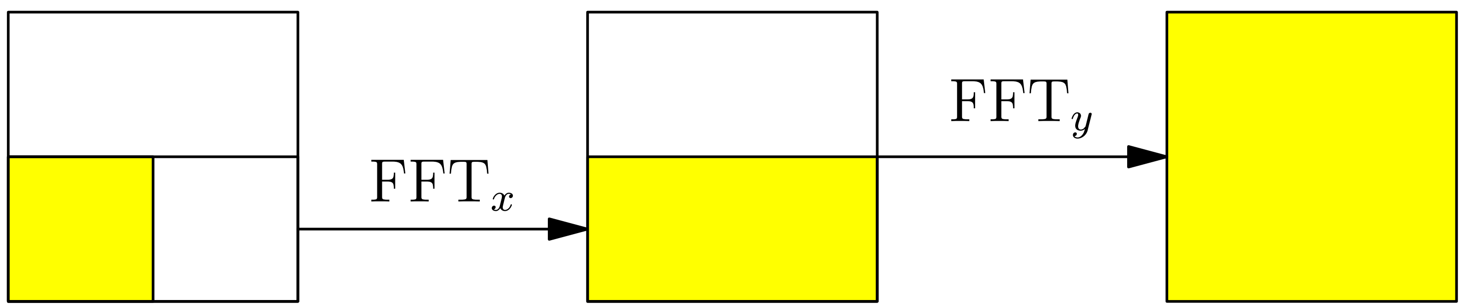
- We recover $f * g$ by taking an inverse padded x-FFT:



- An implicitly padded convolution in 2 dimensions requires only $9N$ padded FFTs,
- and only twice the memory of a cyclic convolution.
- The operation count is $6KN \log N/2$.

Alternatives

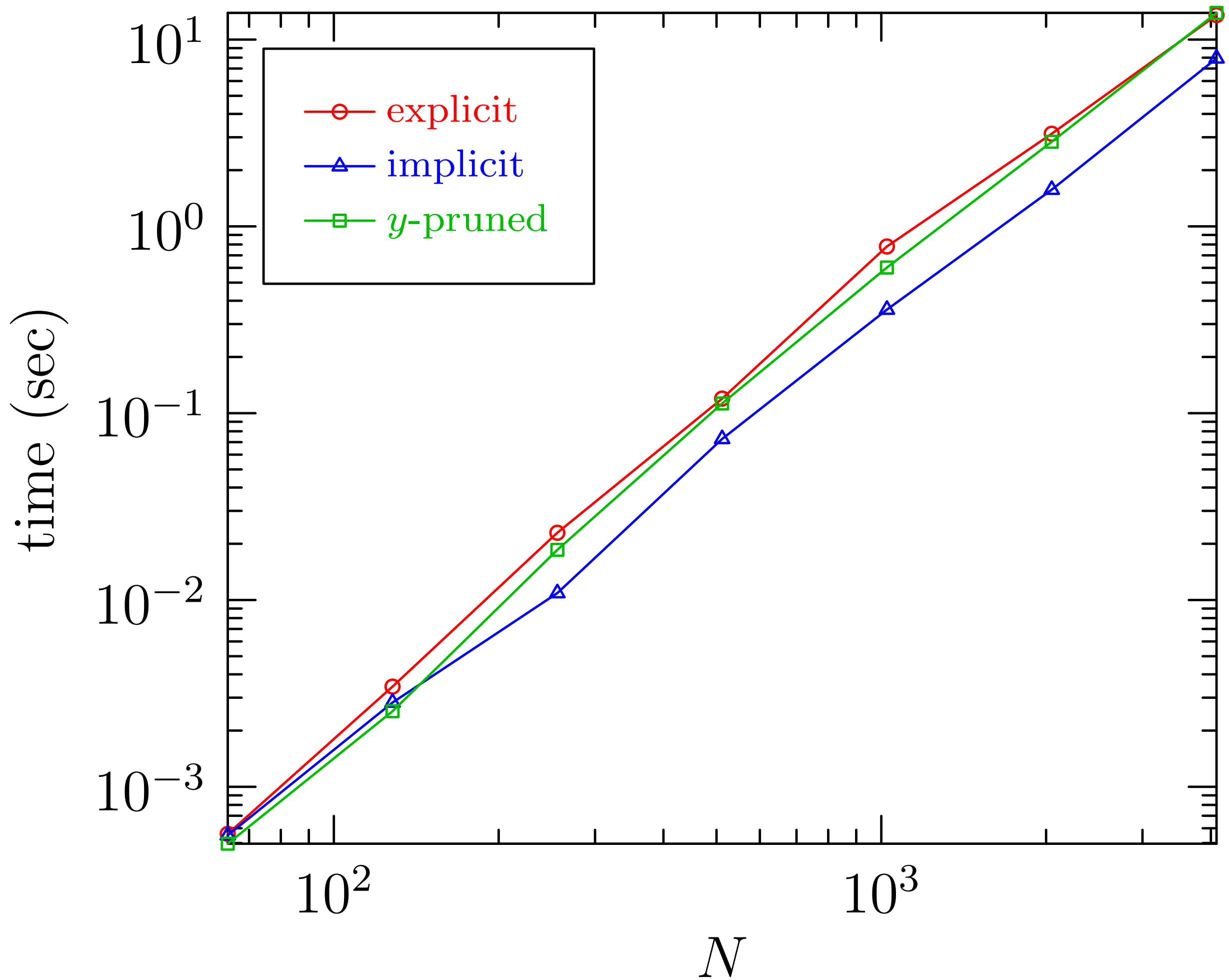
- The memory savings could be achieved more simply by using conventional padded transforms.
- This requires copying data, which is slow.
- Half of the FFTs in the x-direction are on zero-data.
- We can skip ("prune") such transforms:



- This is slower with large data sets due to memory-striding issues.
- Phase-shift dealiasing has the same memory footprint as "1/2" explicit padding.

Implicit Padding in 2D

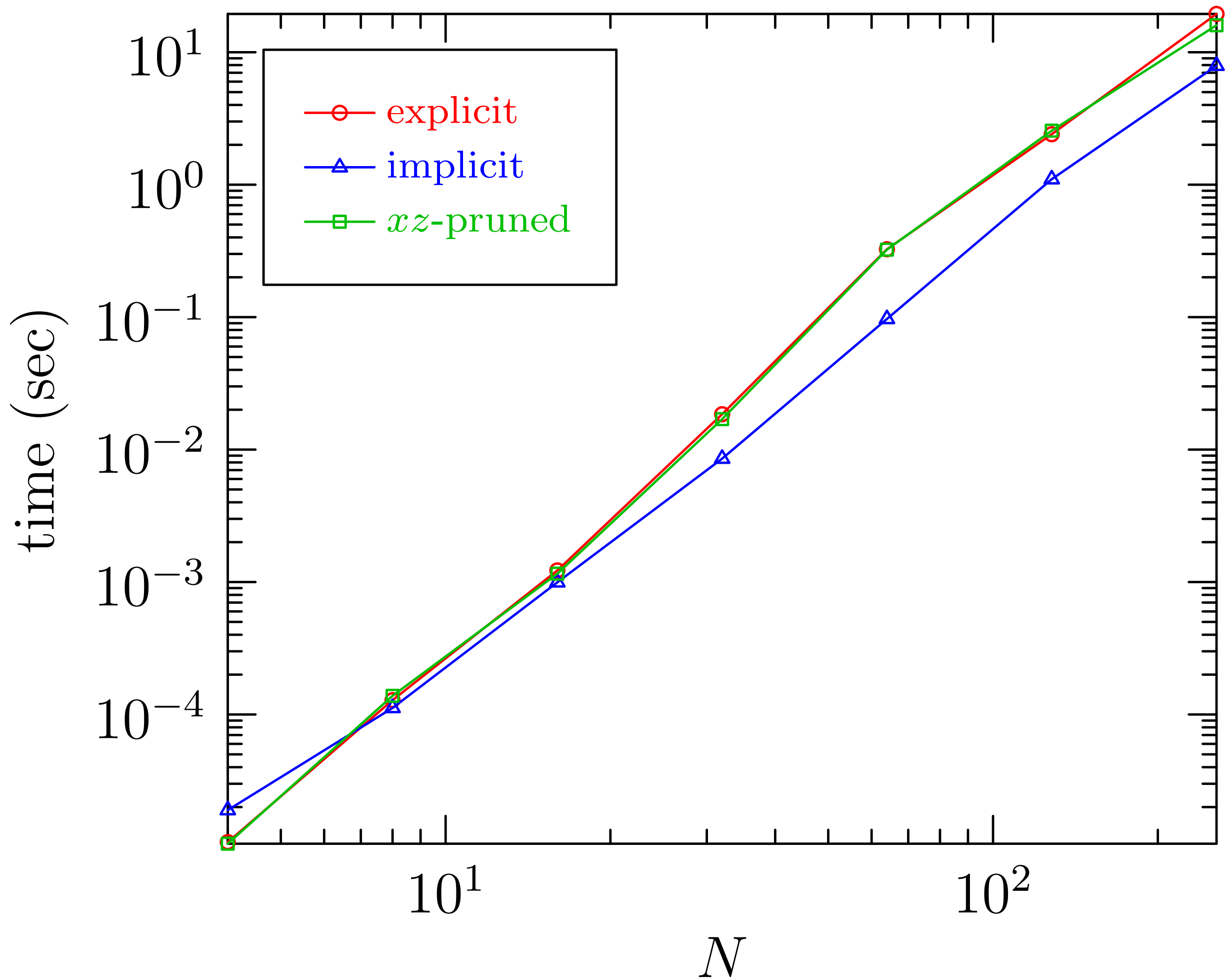
- Implicit padding is faster in two dimensions:



- And uses half the memory of explicit padding.

Implicit Padding in 3D

- The algorithm is easily extended to three dimensions:



- Implicit padding uses $1/4$ the memory of explicit padding in 3D.

Centered Hermitian Data

- The input f is *centered* if $\{f_n\}_{n=-N/2+1}^{N/2-1} \iff \{F_k\}_{k=-N/2+1}^{N/2-1}$.
- If $\{f_n\}$ is real-valued, then $\mathcal{F}(f)$ is *Hermitian*:

$$F_{-k} = \overline{F}_k$$

- The convolution of the centered arrays f and g is

$$(f * g)_n = \sum_{p=n-N/2+1}^{N/2-1} f_p g_{n-p}.$$

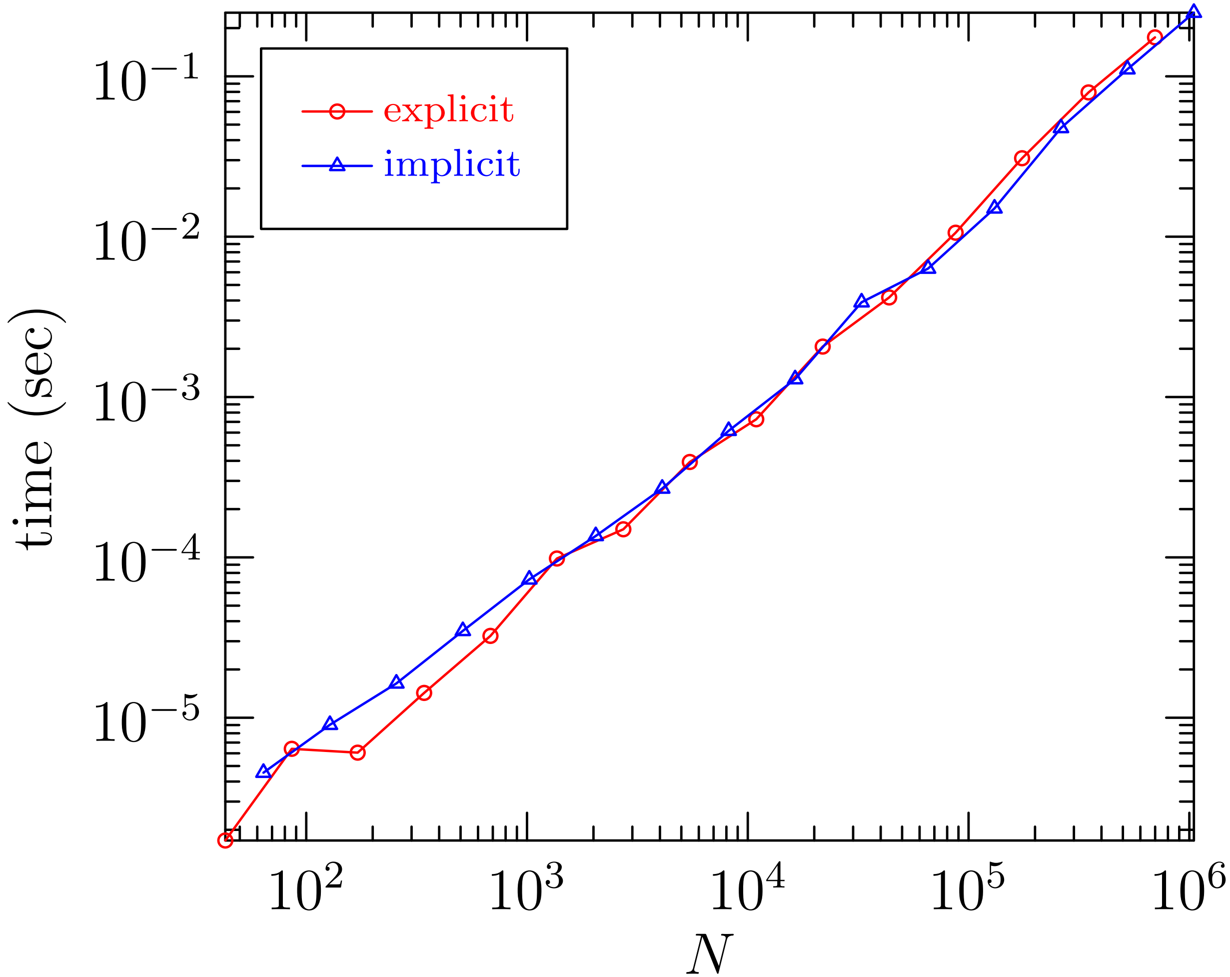
- Padding centered data use a "2/3" rule:

$$\{\widetilde{f}_n\}_{n=-N/2+1}^{N-1} = (f_{-N/2+1}, \ldots, f_0, \ldots, f_{N/2-1}, \underbrace{0, \ldots, 0}_{N/2}).$$

- Phase-shifting is slower than explicit padding for centered data.

Centered Hermitian Data: 1D

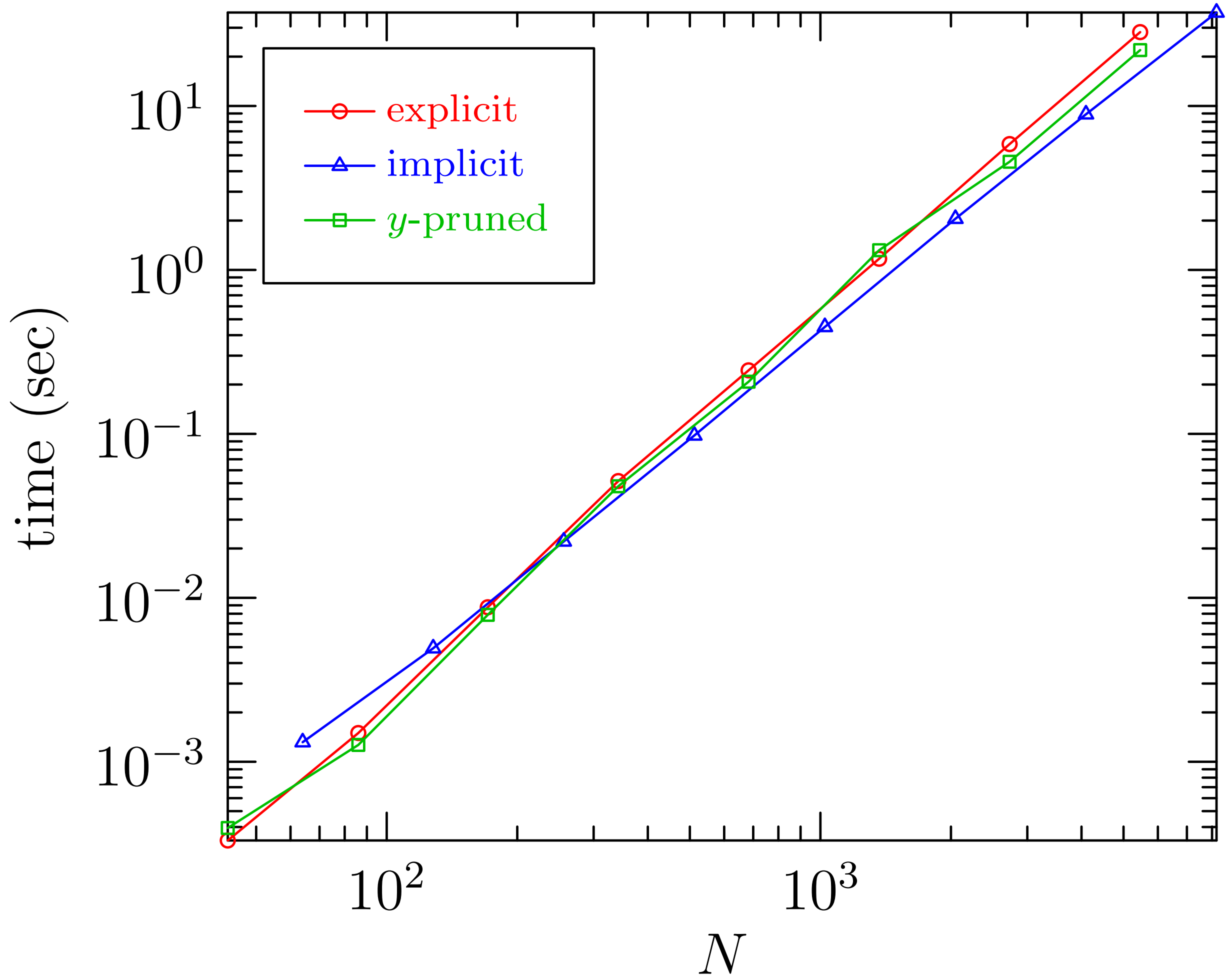
- The 1D implicit convolution is as fast as explicit padding:



- And has a comparable memory footprint.

Centered Hermitian Data: 2D

- Implicit centered convolutions are faster in higher dimensions:



- And uses $(2/3)^{d-1}$ the memory in d dimensions.

Optimal Problem Sizes

- We use convolutions in pseudo-spectral simulations:

$$\partial_t u + u \cdot \nabla u = -\nabla P + \nu \nabla^2 u$$

is advanced in Fourier space, with $u \cdot \nabla u$ calculated in x-space.

- FFTs are faster for highly composite problem sizes:
  - $N = 2^n$, $N = 3^n$, etc., with $N = 2^n$ optimal.
- "2/3" padding: 341, 683, 1365 etc
  - FFTs are of size $N = 512$, 1024, 2048, etc.
- Phase-shift dealiasing: $2^n - 1$
  - FFTs are of length 2^{n-1}.
- Implicit padding: $2^n - 1$.
  - sub-transforms are of size 2^{n-1}.

Ternary Convolutions

- The *ternary convolution* of three vectors f, g, and h is

$$*(f,g,h)_n = \sum_{a,b,c\in\{0,\ldots,N-1\}} f_a\, g_b\, h_c\, \delta_{a+b+c,n}.$$

- Computing the transfer function for $Z_4 = N^3 \sum_{\boldsymbol{j}} \omega^4(x_{\boldsymbol{j}})$ requires computing the Fourier transform of ω^3.

- This requires a centered Hermitian ternary convolution:

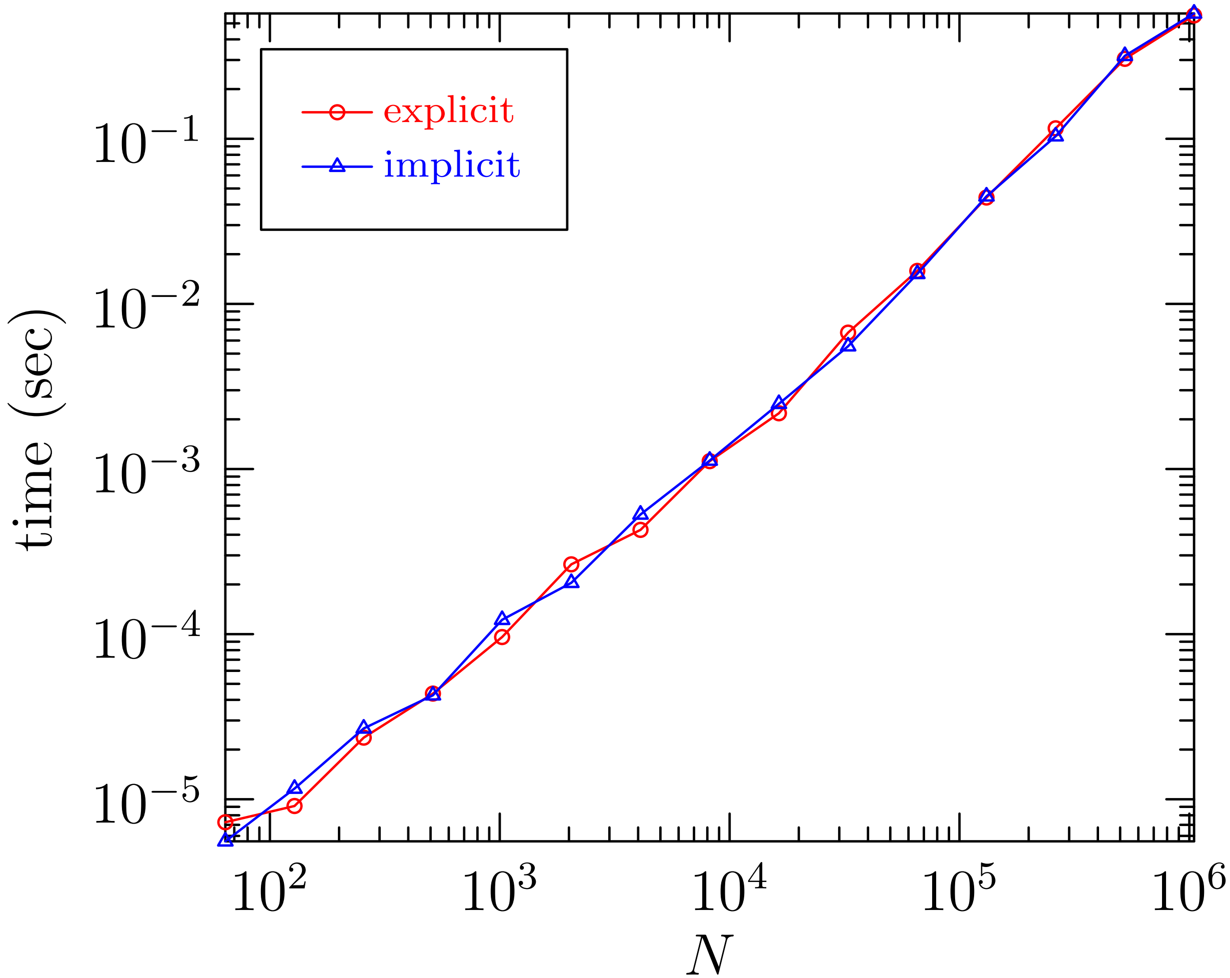
$$*(f,g,h)_n = \sum_{a,b,c\in\{-\frac{N}{2}+1,\ldots,\frac{N}{2}-1\}} f_a\, g_b\, h_c\, \delta_{a+b+c,n}.$$

- Correctly dealiasing requires a "2/4" padding rule.

- Computing Z_4 using 2048×2048 pseudospectral modes simulation retains a maximum physical wavenumber of only 512.

Centered Hermitian Ternary Convolutions: 1D

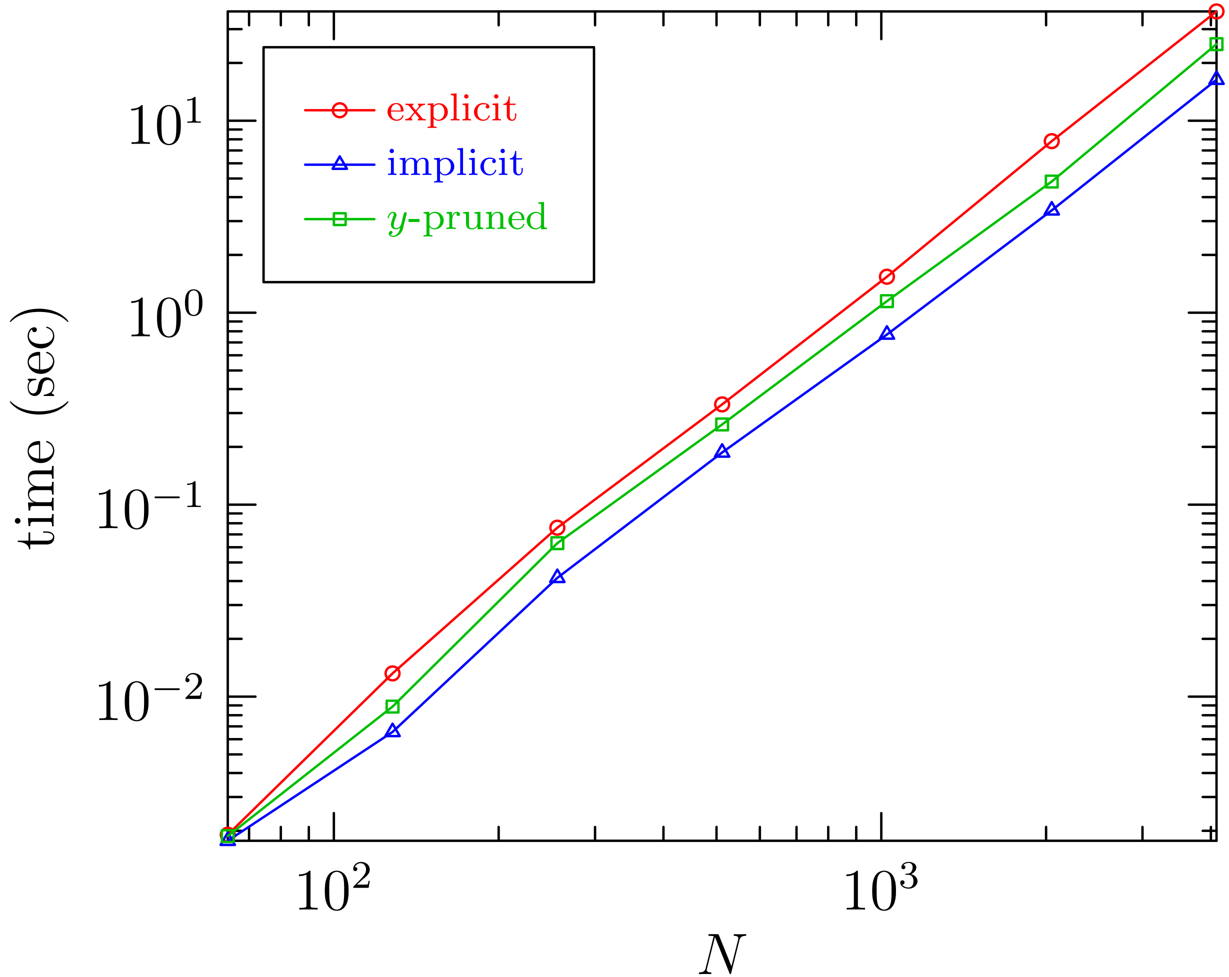
- The 1D implicit ternary convolution is as fast as explicit padding:



- And has a comparable memory footprint.

Centered Hermitian Ternary Convolutions: 2D

- Implicit centered ternary convolutions are faster in higher 2D:



- And use $(1/2)^{d-1}$ the memory in d dimensions.

Conclusion

- Implicitly padded fast convolutions eliminate aliasing errors.
- Implicit padding uses $(p/q)^{d-1}$ the memory of explicit d-dimensional "p/q" padding.
- Computational speedup from increased data locality and pruning FFTs.
- Expanding discontiguously is easier to program.
- "Efficient Dealiased Convolutions without Padding" submitted to SIAM Journal on Scientific Computing.
- A `C++` implementation under the LGPL is available at `http://fftwpp.sourceforge.net/`.
- Fastest Fourier Transform in the West (`http://fftw.org/`) provides sub-transforms.

References

[Cooley & Tukey 1965] J. W. Cooley & J. W. Tukey, Mathematics of Computation, **19**:297, 1965.

[Gauss 1866] C. F. Gauss, “Nachlass: Theoria interpolationis methodo nova tractata,” in *Carl Friedrich Gauss Werke*, volume 3, pp. 265–330, Königliche Gesellschaft der Wissenschaften, Göttingen, 1866.

[Patterson Jr & Orszag 1971] G. S. Patterson Jr & S. A. Orszag, Physics of Fluids, **14**:2538, 1971.