The Fastest Convolution in the West

Malcolm Roberts and John Bowman

University of Alberta

2011-02-15

www.math.ualberta.ca/∼mroberts

Outline

- Convolution
  - Definition
  - Applications
- Fast Convolutions
- Dealiasing Convolutions
  - Zero-padding
  - Phase-shift dealiasing
  - Implicit Padding
- Convolutions in Higher Dimensions
- Centered Hermitian Convolutions
- Ternary Convolutions

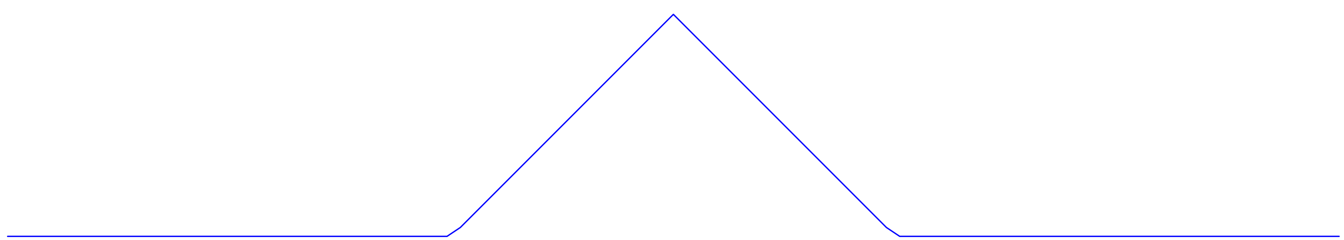
Convolutions

- 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*:

$$(F * G)_k = \sum_{\ell=0}^{k} F_\ell G_{k-\ell}.$$

- Calculating $\{(F * G)_k\}_{k=0}^{N-1}$ directly takes $\mathcal{O}(N^2)$ operations.

- This method is not numericaly robust.

Fast Convolutions

- The unnormalized backward Fourier transform is

$$\{f_n\}_{n=0}^{N-1} = \mathcal{F}^{-1}[F] = \sum_{k=0}^{N-1} \zeta_N^{kn} F_k$$

where $\zeta_N = e^{-2\pi i/N}$ is the N^{th} root of unity.

- The forward transform is

$$\{F_k\}_{k=0}^{N-1} = \mathcal{F}[f] = \frac{1}{N} \sum_{n=0}^{N-1} \zeta_N^{-kn} f_n$$

- This transform relies on the identity

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

Fast Convolutions

- Convolutions are multiplications when Fourier-transformed:

$$\begin{aligned}\mathcal{F}^{-1}[F * G] = \sum_{j=0}^{N-1} f_j g_j \zeta_N^{-jk} &= \sum_{j=0}^{N-1} \zeta_N^{-jk} \left(\sum_{p=0}^{N-1} \zeta_N^{jp} F_p \right) \left(\sum_{q=0}^{N-1} \zeta_N^{jq} G_q \right) \\ &= \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} F_p G_q \sum_{j=0}^{N-1} \zeta_N^{(-k+p+q)j} \\ &= N \sum_{s} \sum_{p=0}^{N-1} F_p G_{k-p+sN}.\end{aligned}$$

- The terms with $s \neq 0$ are aliases; this is a cyclic convolution:

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

Dealiasing via Explicit Zero-Padding

- If we pad F and G with zeroes, we recover the linear convolution:

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

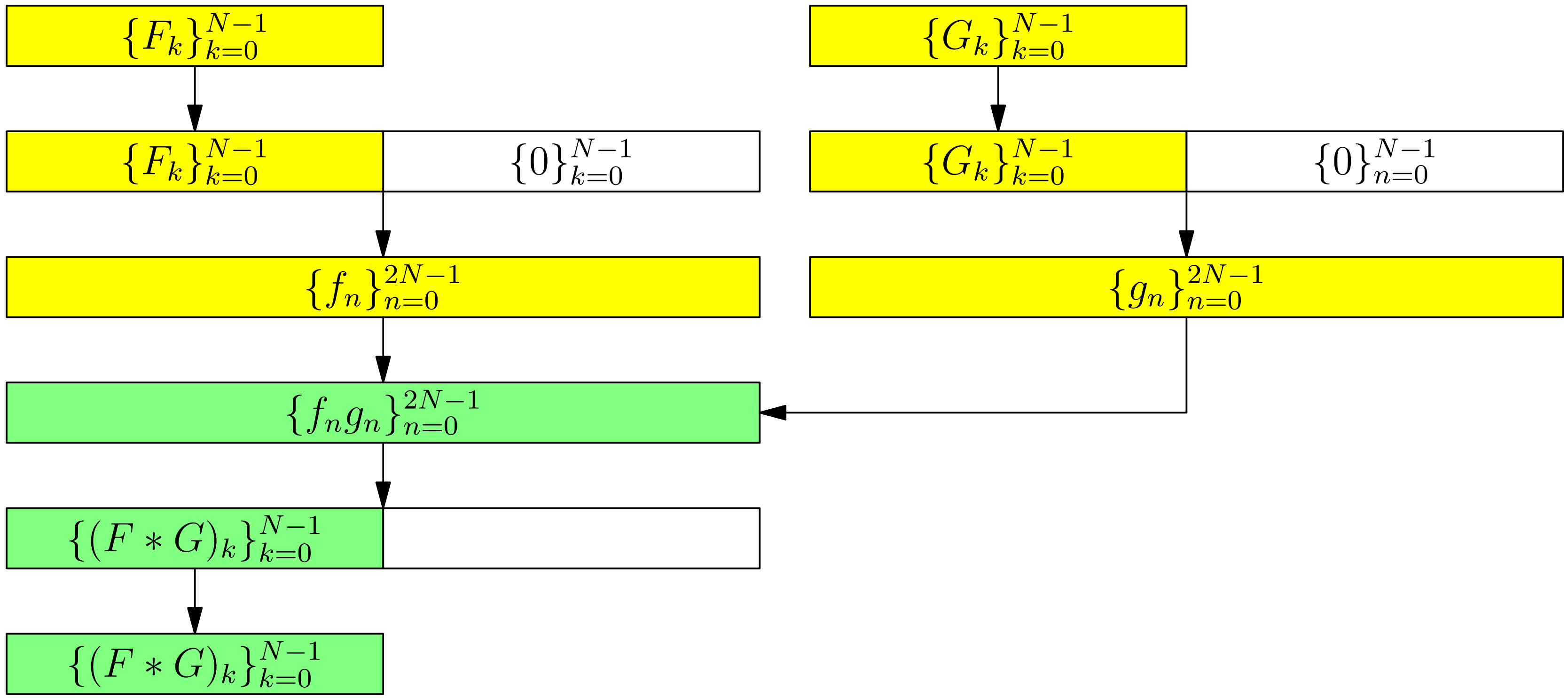
- Then,

$$(\widetilde{F} *_{2N} \widetilde{G})_k = \sum_{\ell=0}^{2N-1} \widetilde{F}_{\ell (\mathrm{mod}\, 2N)} \widetilde{G}_{(k-\ell)(\mathrm{mod}\, 2N)},$$

$$= \sum_{\ell=0}^{N-1} F_\ell \widetilde{G}_{(k-\ell)(\mathrm{mod}\, 2N)},$$

$$= \sum_{\ell=0}^{k} F_\ell G_{k-\ell}.$$

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 **?** is

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

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

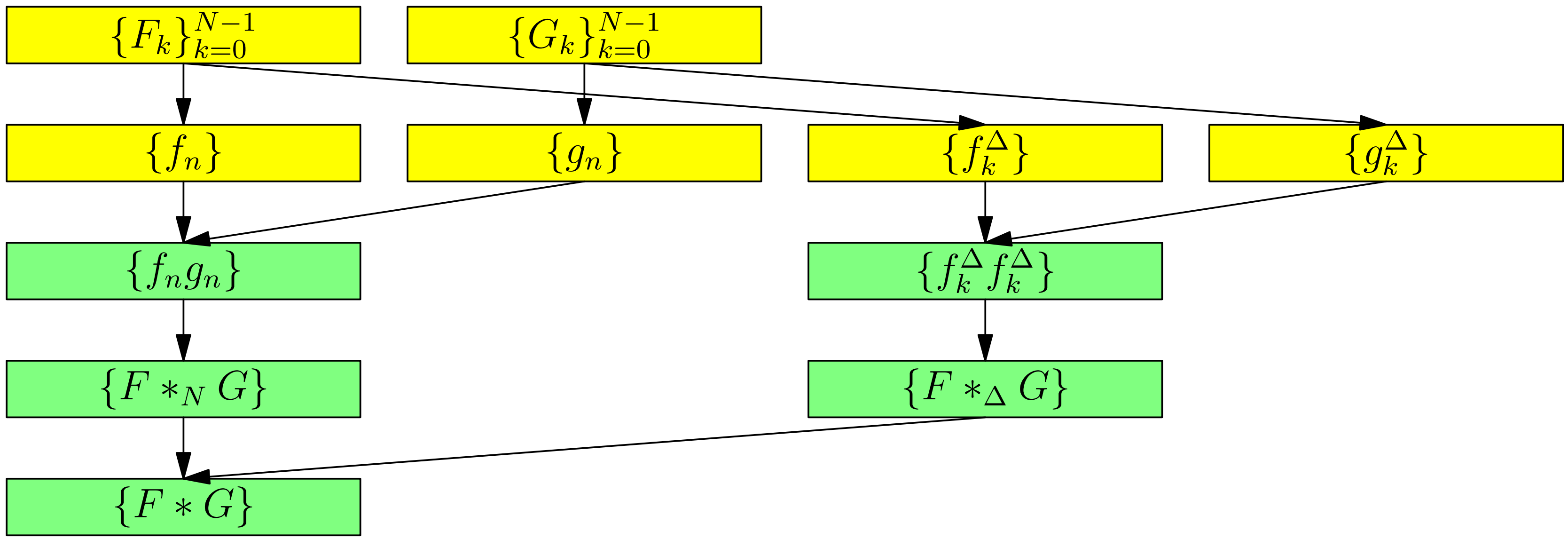
$$(F *_{\Delta} G)_k \doteq \mathcal{F}^{\Delta}\big(f^{\Delta} g^{\Delta}\big) = \sum_{\ell=0}^{k} F_\ell G_{k-\ell} - \sum_{\ell=k+1}^{N-1} F_\ell G_{k-\ell+N},$$

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

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

- The discrete Fourier transform is a sum:

$$\mathcal{F}^{-1}(F)_k = \sum_{k=0}^{2N-1} \zeta_{2N}^{kn} F_k.$$

- Since $F_k = 0$ if $k \geq N$, this is just

$$\mathcal{F}^{-1}(F)_n = \sum_{k=0}^{N-1} \zeta_{2N}^{kn} F_k.$$

- This is not a Fourier transform: the FFT algorithm doesn't apply.

Implicit Padding

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

$$f_{2n} = \sum_{k=0}^{N-1} \zeta_{2N}^{k2n} F_k = \sum_{k=0}^{N-1} \zeta_N^{kn} F_k,$$

$$f_{2n+1} = \sum_{k=0}^{N-1} \zeta_{2N}^{k(2n+1)} F_k = \sum_{k=0}^{N-1} \zeta_N^{kn} \left(F_k \zeta_{2N}^{k}\right),$$

which *are* Fourier transforms.

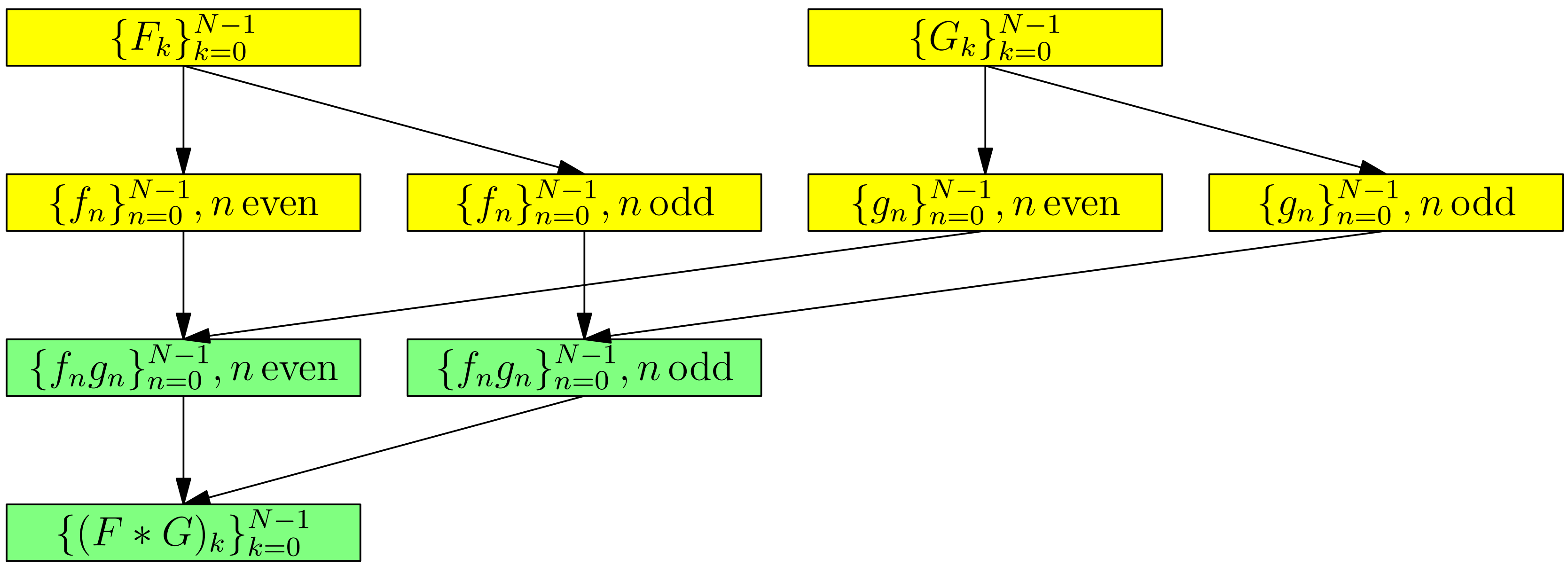
- The inverse is the sum of two Fourier transforms:

$$F_k = \frac{1}{N}\left(\sum_{n=0}^{N-1} \zeta_N^{-kn} f_{2n} + \zeta_{2N}^{k} \sum_{k=0}^{N-1} \zeta_N^{-kn} f_{2n+1}\right).$$

Implicit Padding

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

- However, the extra memory need not be contiguous: this will be shown to be quite advantageous.

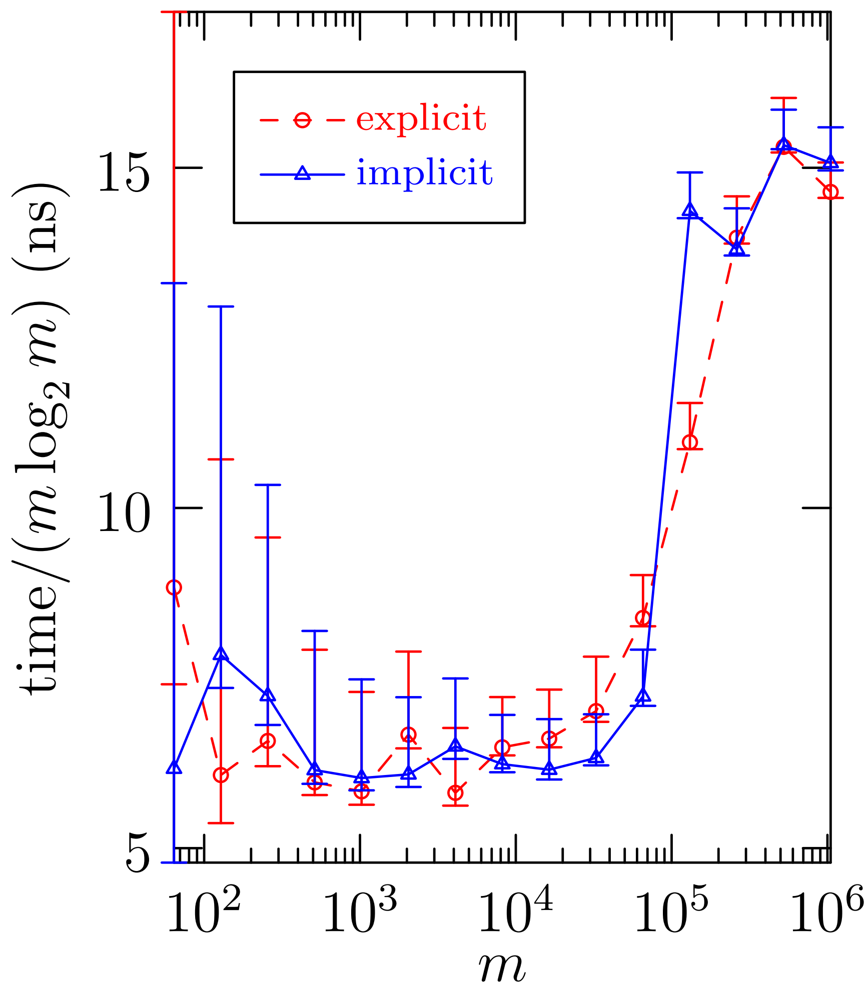


- The computational complexity is $6KN\log_2(N/2)$.

- The numerical error is similar to explicit padding.

Implicit Padding: speed

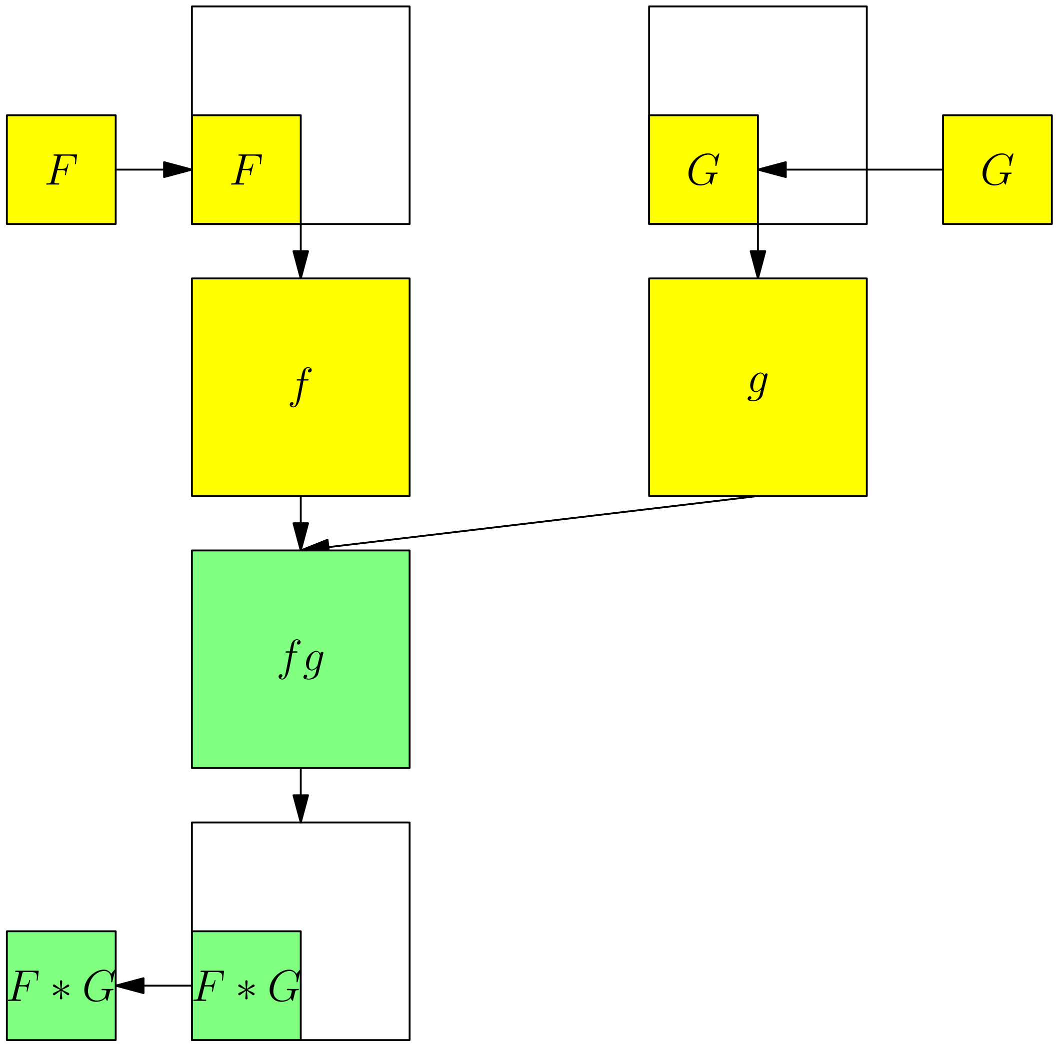
- The algorithms are comparable in speed:



- Ours is much more complicated.

Convolutions in Higher Dimensions

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



Convolutions in Higher Dimensions

- Notice that 3/4 of the transformed arrays are zero.

- It is possible to skip these transforms

  i.e. use a pruned FFT.

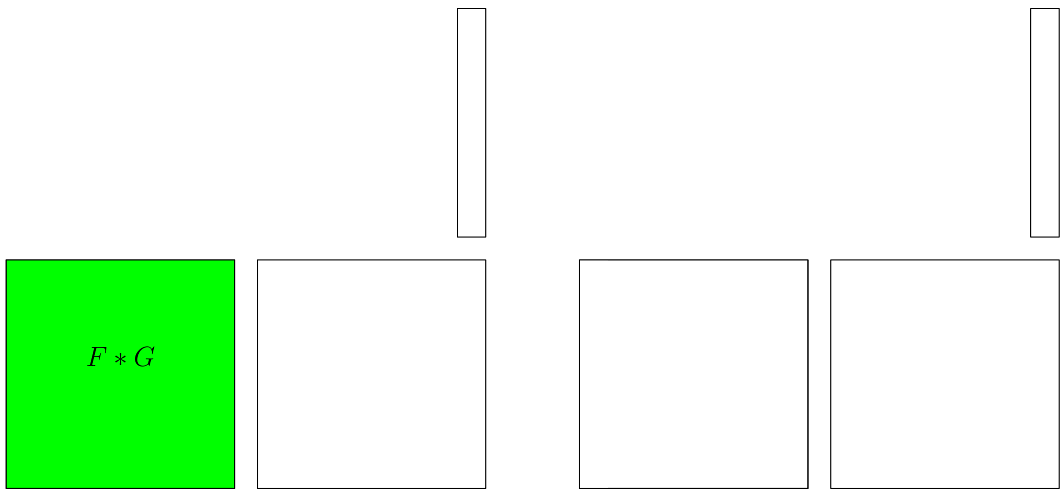
- In the absence of a specially optimized routine for pruned FFTs, it can be faster to simply transform the entire array.

Implicit Convolutions in Higher Dimensions

- One can perform an implicitly-padded 2D convolution by first performing a backward transform in the x-direction,

  then performing an implicit 1D convolution in the y-direction,

  and then performing a forward transform in the x-direction.



- An implicitly padded convolution in 2 dimensions requires only $9N$ padded FFTs,

  and only twice the memory of a cyclic convolution.

Alternatives

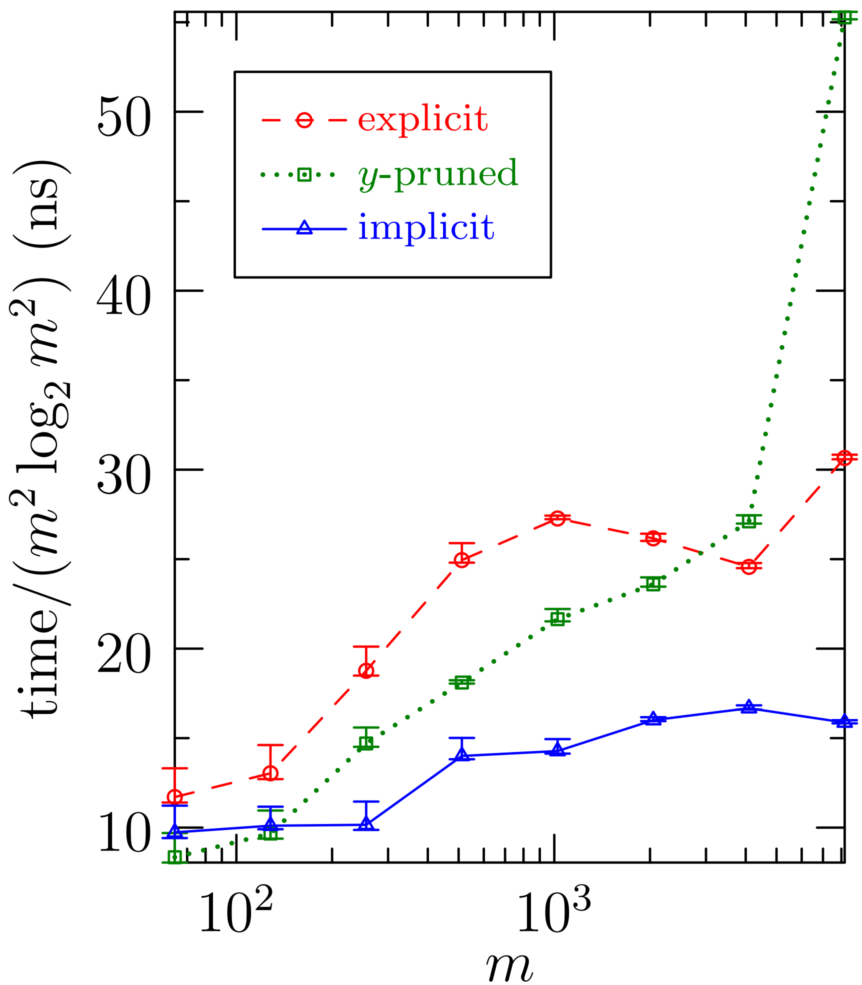
- The memory savings could be achieved more simply by using conventional padded transforms.

- This requires copying data, which is slow.

- Phase-shift dealiasing has the same memory footprint as "1/2" explicit padding.

Implicit Padding in 2D

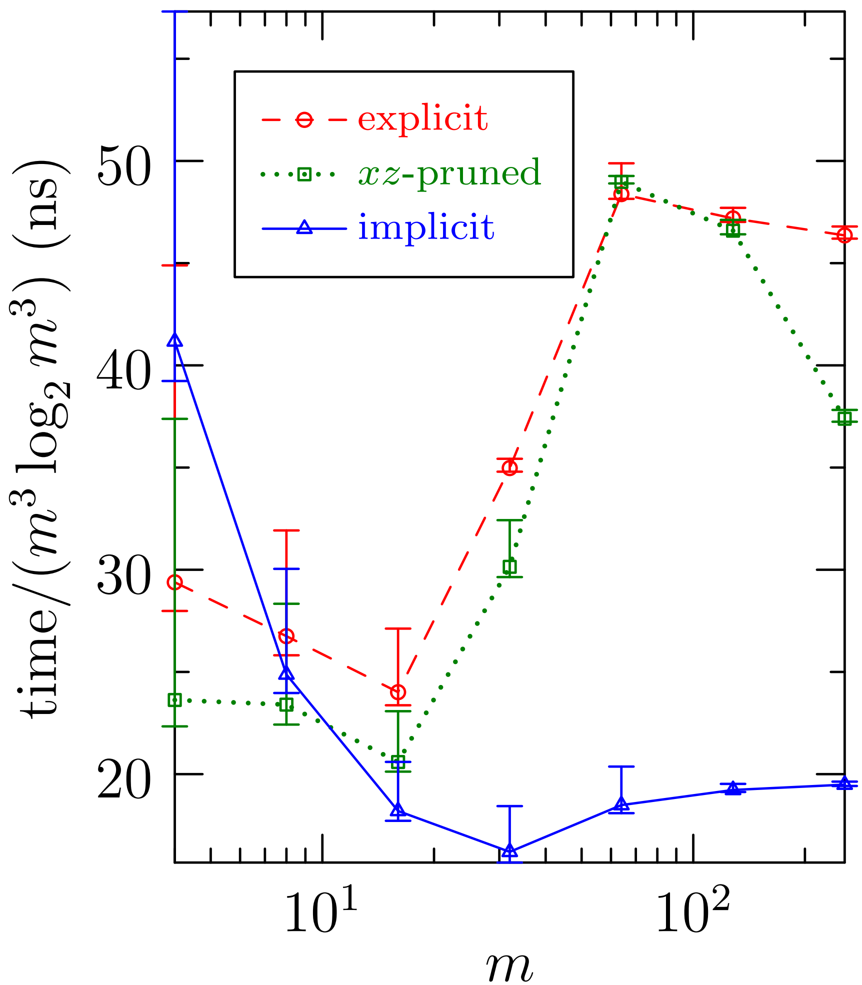
- Implicit padding is faster in two dimensions:



- 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 $\{F_k\}_{k=-N/2+1}^{N/2-1} \iff \{f_n\}_{n=-N/2+1}^{N/2-1}$.
- If $\{f_n\}$ is real-valued, then F is *Hermitian*:

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

- The convolution of the centered arrays f and g is

$$(F * G)_k = \sum_{\ell=k-N/2+1}^{N/2-1} F_\ell G_{k-\ell}.$$

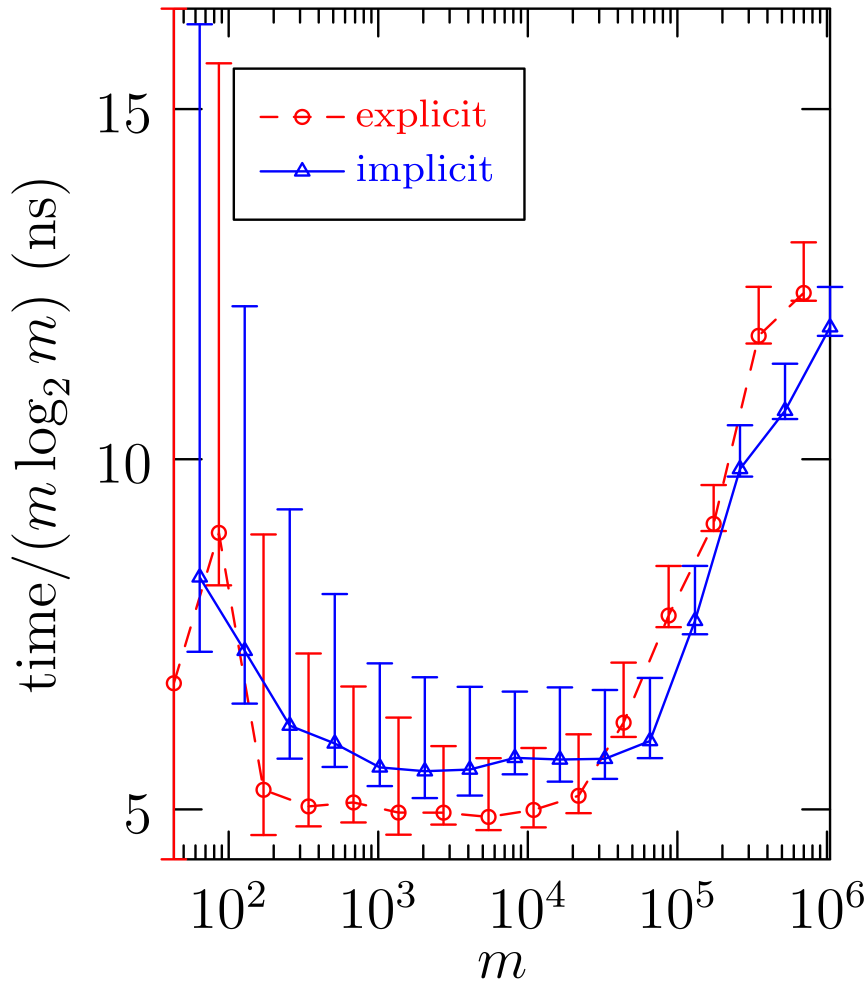
- Padding centered data use a "2/3" rule:

$$\{\widetilde{F}_k\}_{k=-N/2+1}^{N-1} = (F_{-N/2+1}, \ldots, F_0, \ldots, F_{N/2-1}, \underbrace{0, \ldots, 0}_{N/2}).$$

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

Centered Hermitian Data: 1D

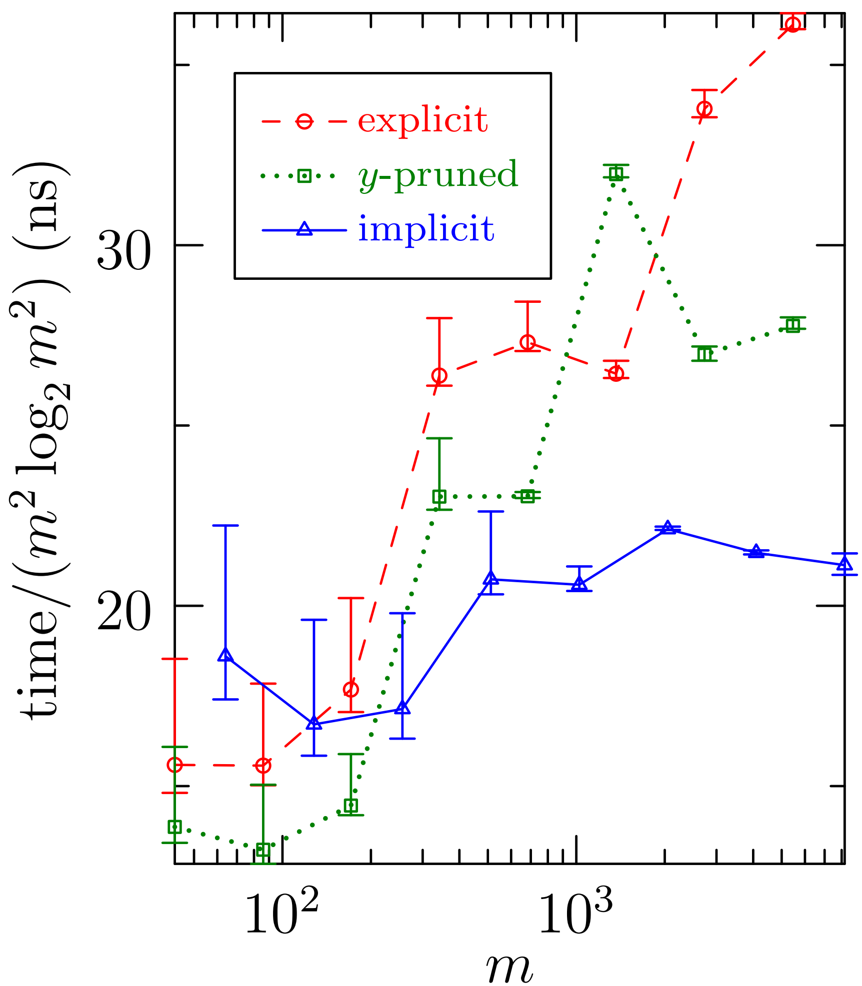
- The 1D implicit convolution is as fast as explicit padding:



- 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.

Example: 2D pseudospectral Navier–Stokes

- These routines are available in the open-source package **`FFTW++`**
- We need to compute:

$$\frac{\partial \omega}{\partial t} = -\boldsymbol{u}\boldsymbol{\cdot}\boldsymbol{\nabla}\omega = -(\hat{\boldsymbol{z}}\times\boldsymbol{\nabla}\nabla^{-2}\omega)\boldsymbol{\cdot}\boldsymbol{\nabla}\omega,$$

which appears in Fourier space as

$$\frac{\partial \omega_{\boldsymbol{k}}}{\partial t} = \sum_{\boldsymbol{k}=\boldsymbol{p}+\boldsymbol{q}} \frac{p_x q_y - p_y q_x}{q^2}\omega_{\boldsymbol{p}}\omega_{\boldsymbol{q}}.$$

- The right-hand side of this equation may be computed as

$$\texttt{ImplicitHConvolution2}(ik_x\omega, ik_y\omega, ik_y\omega/k^2, -ik_x\omega/k^2).$$

Optimal Problem Sizes

- 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

$$*(F, G, H)_k = \sum_{a,b,c\in\{0,\ldots,N-1\}} F_a\, G_b\, H_c\, \delta_{a+b+c,n}.$$

- 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:

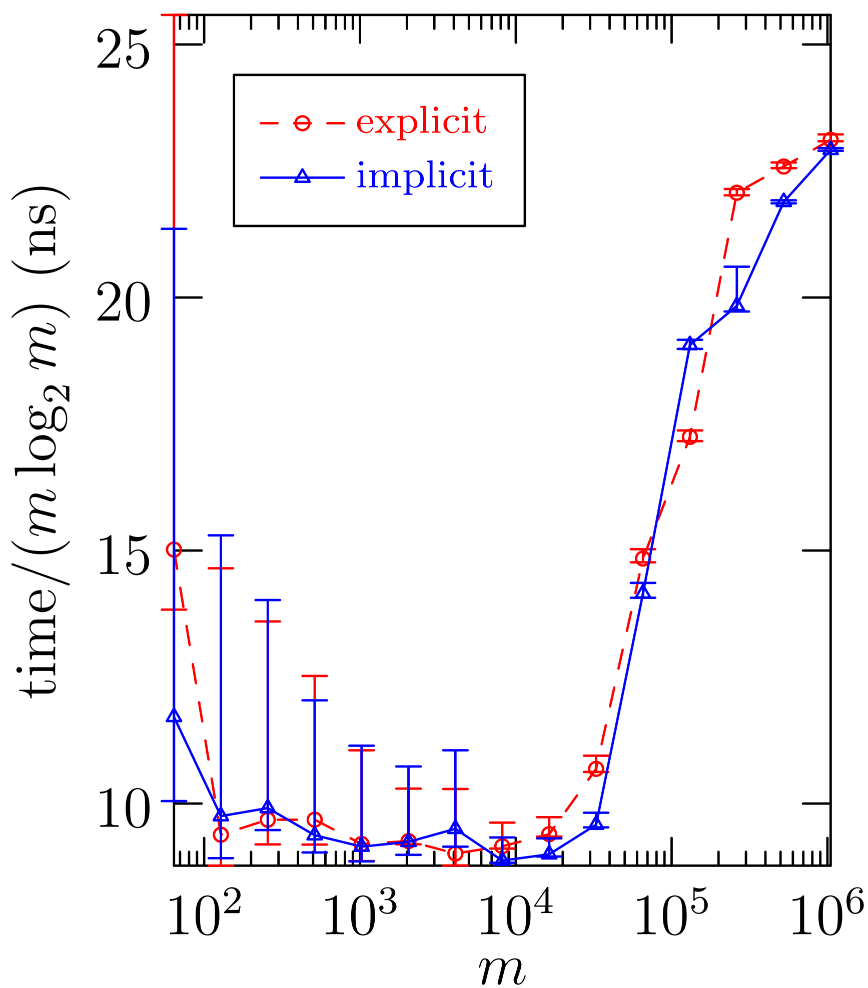
$$*(F, G, H)_k = \sum_{a,b,c\in\{-\frac{N}{2}+1,\ldots,\frac{N}{2}-1\}} F_a\, G_b\, H_c\, \delta_{a+b+c,n}.$$

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

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

Centered Hermitian Ternary Convolutions: 1D

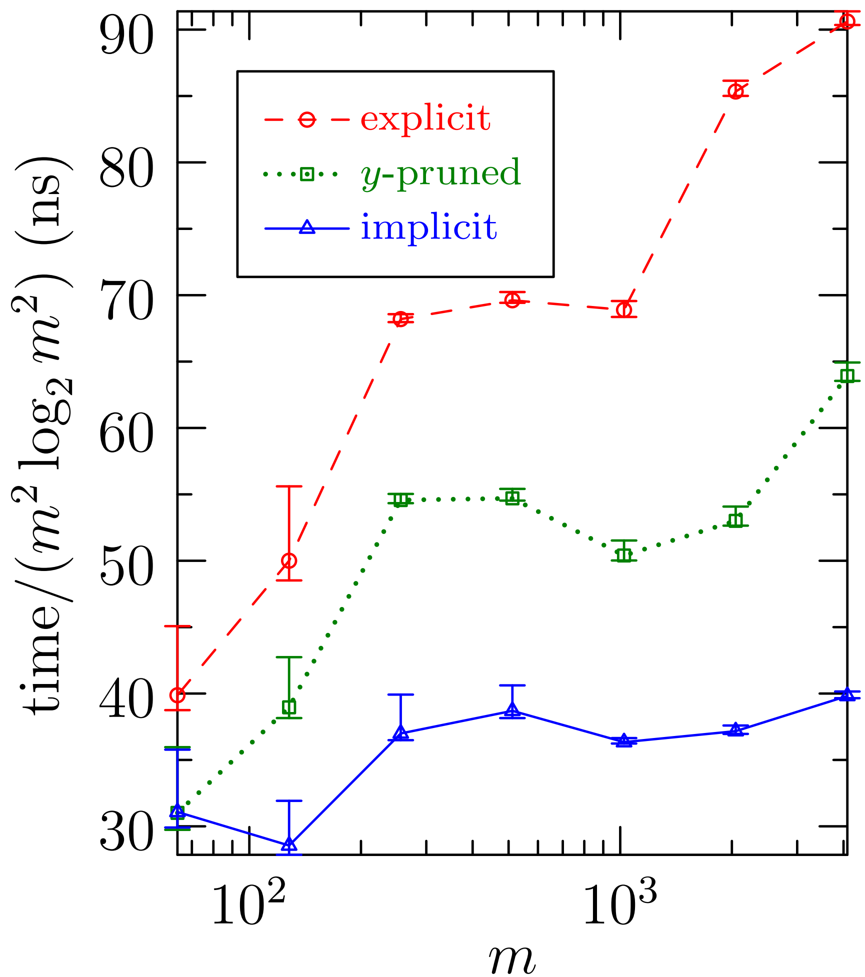
- The 1D implicit ternary convolution is as fast as explicit padding:



- 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.

FFTW++

- A `C++` implementation, (`FFTW++`, LGPL) is available at `http://fftwpp.sourceforge.net/`.
- Fastest Fourier Transform in the West (`http://fftw.org/`) provides sub-transforms.
- Future work: parallelize `FFTW++`.
- Available in `FFTW++`:
  - Non-centered convolutions in 1D, 2D, and 3D,
  - Centered Hermitian convolutions in 1D, 2D, and 3D,
  - Centered Hermitian ternary convolutions in 1D, 2D.

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 skipping a bit-reversal in the FFT and pruning FFTs efficiently.

- Expanding discontiguously is easier to program.

- *Efficient Dealiased Convolutions without Padding*, SIAM Journal on Scientific Computing, **33**, 386–406 (2011).