The Fastest Convolution in the West

Malcolm Roberts (University of Alberta)

Acknowledgements: John Bowman

May 20, 2010

www.math.ualberta.ca/∼mroberts

Outline

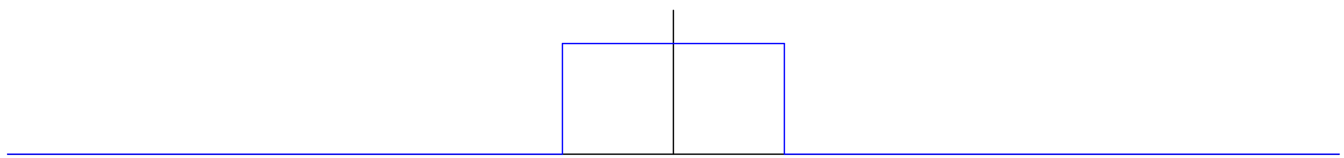
- Convolution
  - Definition
  - Applications
- Fast Convolutions
  - The Convolution Theorem
- Aliasing Errors
  - Zero-padding
  - Phase-shift dealiasing
- Implicit Padding
  - In one dimension
  - In higher dimensions
- Hermitian Convolutions

Convolutions

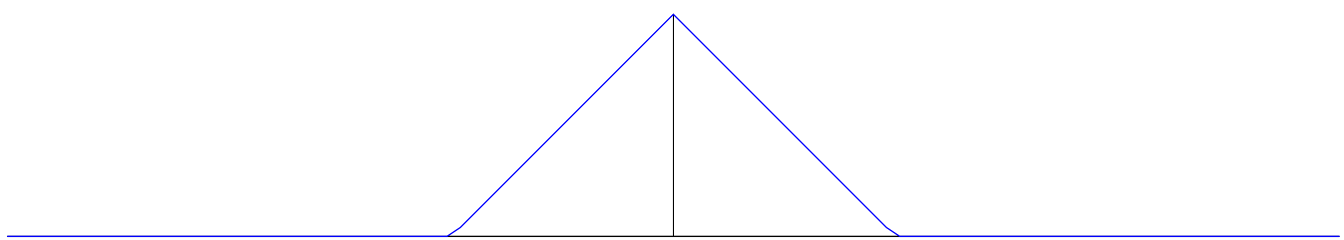
- The convolution of the functions f and g is

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

- 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 a multiplications in Fourier space:

$$\mathcal{F}(f * g) = \mathcal{F}(f)\,\mathcal{F}(g)$$

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

- 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*,

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

  where the vectors f and g have period N.
- The difference between linear and cyclic convolutions,

$$\sum_{m=0}^{N-1} f_m g_{n-m} = \sum_{m=0}^{n} f_m g_{n-m} + \sum_{m=n+1}^{N-1} f_m g_{n-m+N},$$

  is called the *aliasing error.*

Dealiasing via Explicit Zero-Padding

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

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

- Convolving these padded arrays takes $6KN \log_2 2N$ operations,
- and 2^d times the memory, where d is the dimension.
- Memory size and CPU speed have increased much faster than memory bandwidth; this is the *von-Neumann bottleneck*.
- Explicit zero-padding seems wasteful.

Phase-shift Dealiasing

- Another possibility is to use a phase shift [Canuto *et al.* 2006].
- Define the shifted Fourier transform of f to be

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

- Then, setting $\Delta = \pi/2$, 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.

- Thus, we can calculate $f * g$ by from two periodic convolutions.
- This requires $6KN \log_2 N$ operations.

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 FFT, and cannot be done in $\mathcal{O}(N \log_2 N)$.

Implicit Padding

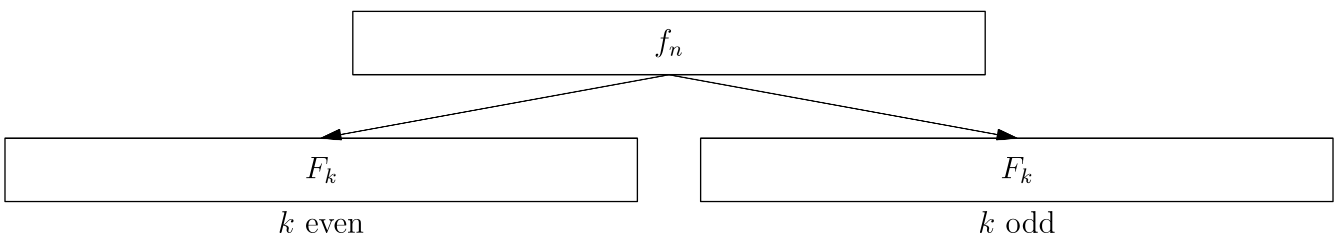
- However, if we calculate even and odd terms separately, we get

$$\mathcal{F}(f)_{2k} = \sum_{n=0}^{N-1} e^{\frac{2\pi i}{N}kn} f_n, \quad \mathcal{F}(f)_{2k+1} = e^{\frac{ik}{2N}} \sum_{n=0}^{N-1} e^{\frac{2\pi i}{N}kn} f_n,$$

  which *are* FFTs.

- The computational complexity is $6KNlog_2N/2$.

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

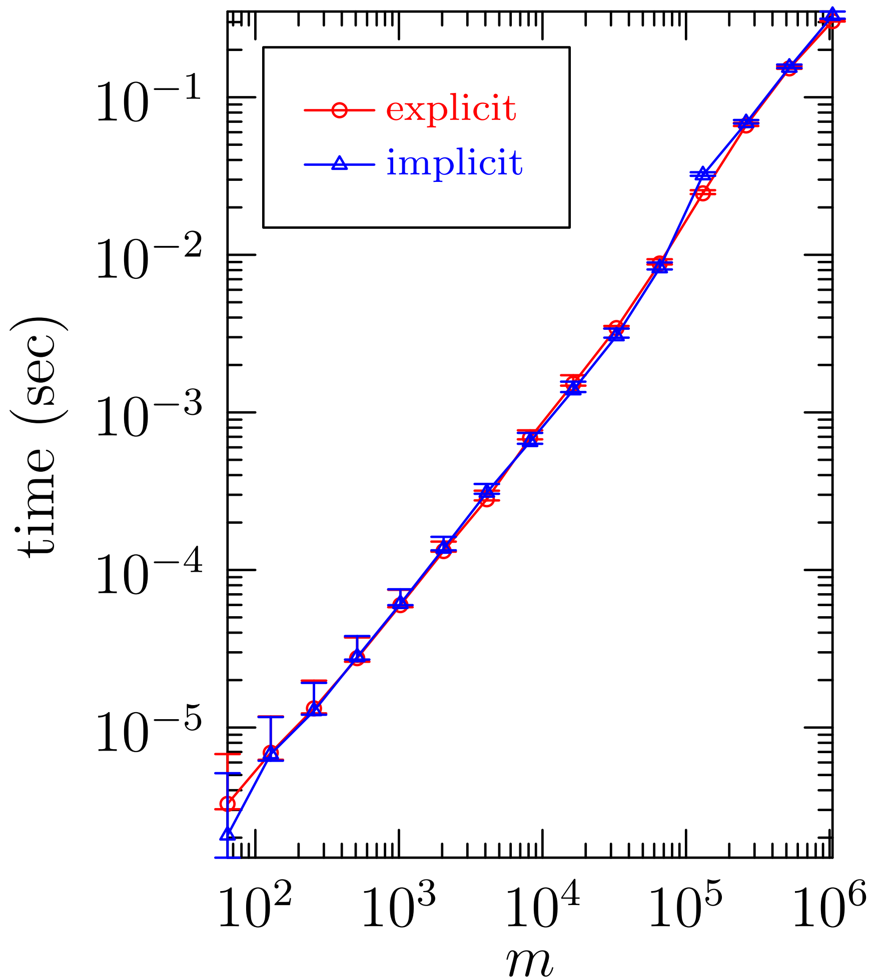


- There is one advantage:

  the work buffer is separate from the data buffer.

Implicit Padding: speed

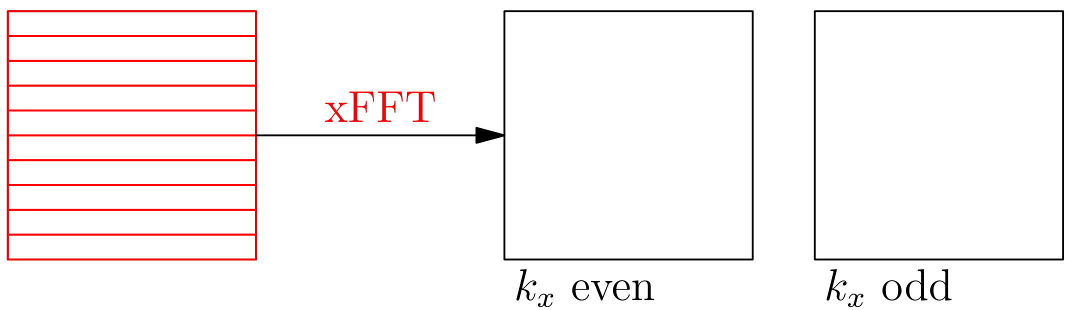
- The algorithms are comparable in speed:



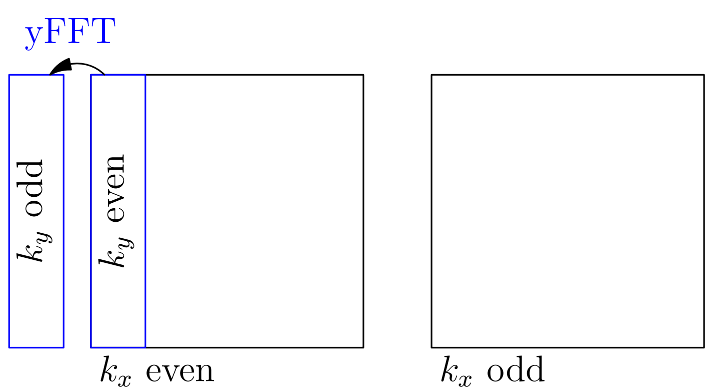
- Ours is much more complicated.

Implicit Convolutions in Higher Dimensions

- 2D fast convolutions involve a series of FFTs, once for each dimension.
- The first FFT produce needs a separate (but non-contiguous) array:

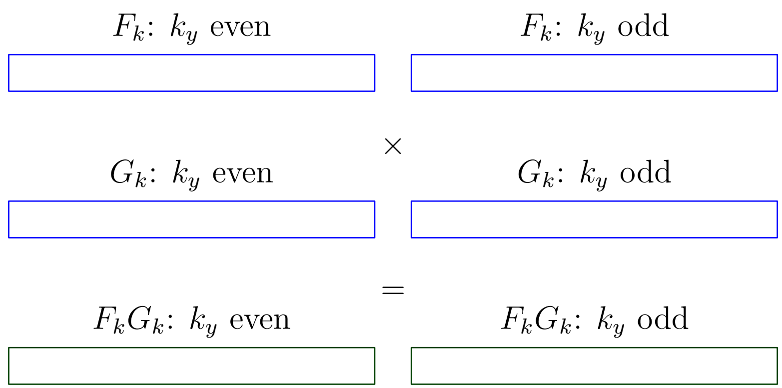


- y-FFTs are done using a 1D work array:

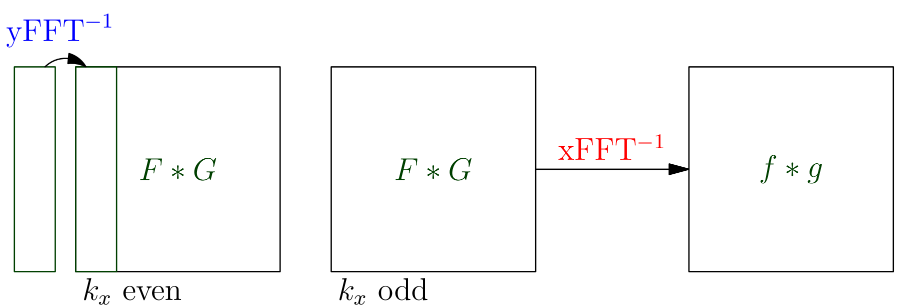


Implicit Convolutions in Higher Dimensions

- The transformed arrays are multiplied:



- Once we have F_kG_k, we take the inverse transform to get $f * g$:



- The resulting algorithm needs half the memory.
- The operation count is $6KN \log N/2$.

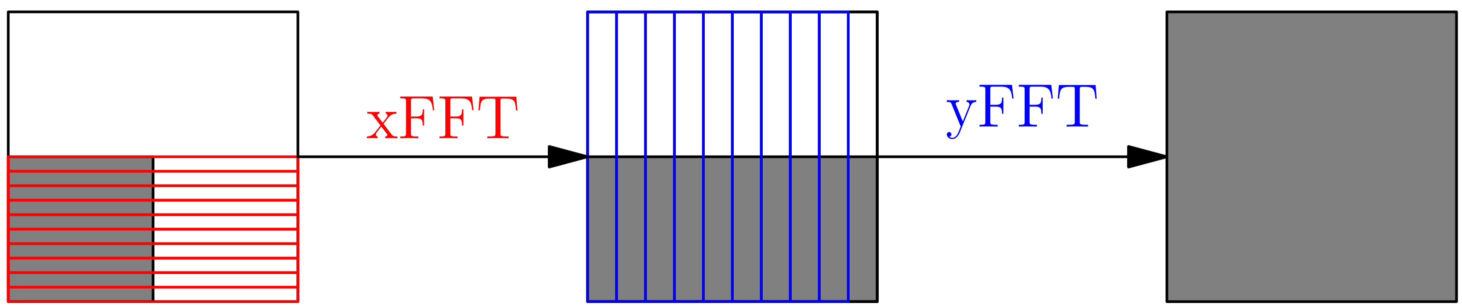
Alternatives

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

  However, this requires copying more data, which is slow.

- Pruning: note that half of the FFTs in the x-direction are on zero-data.

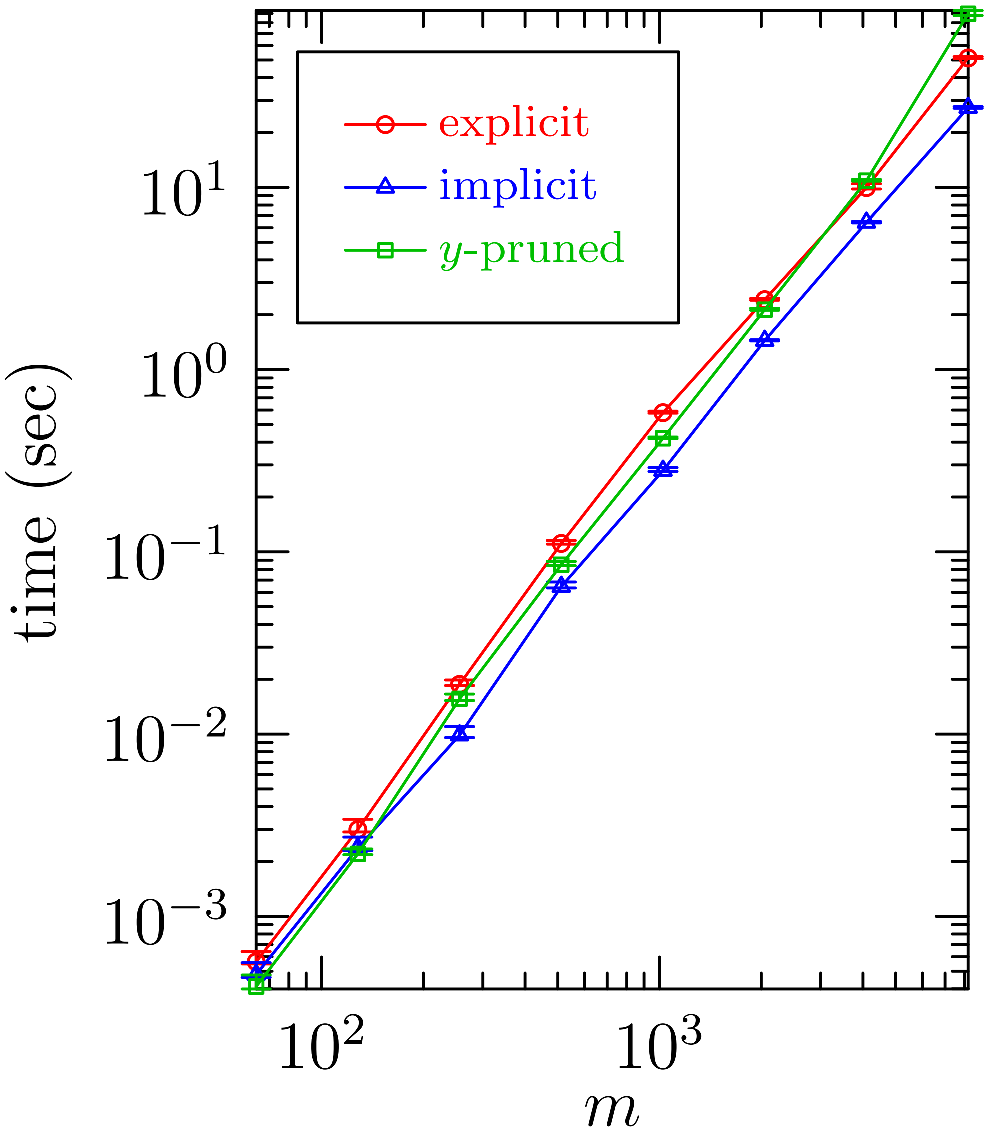
  We can skip such transforms:



  This is actually slower for large data sets due to memory-striding issues.

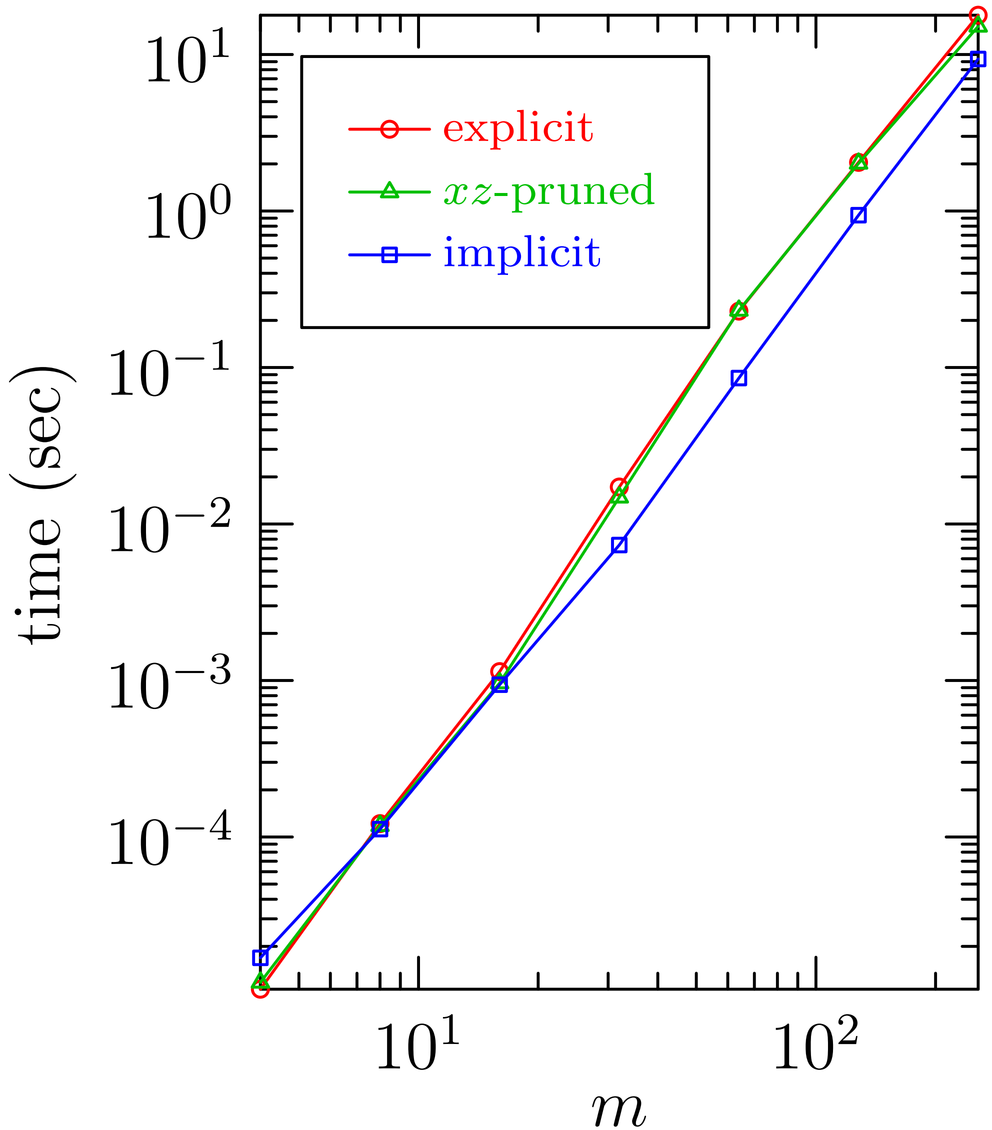
Implicit Padding in Higher Dimensions

- Implicit padding is faster in two dimensions:



Implicit Padding in Higher Dimensions

- The algorithm is easily extended to three dimensions:



Hermitian Data

- If $\{f_n\}_{n=0}^{N-1}$ is real-valued, then

$$\mathcal{F}(f) = \{F_k\}_{k=-N/2}^{N/2}$$

  and $F_{-k} = \overline{F}_k$. Such data is called *Hermitian*.

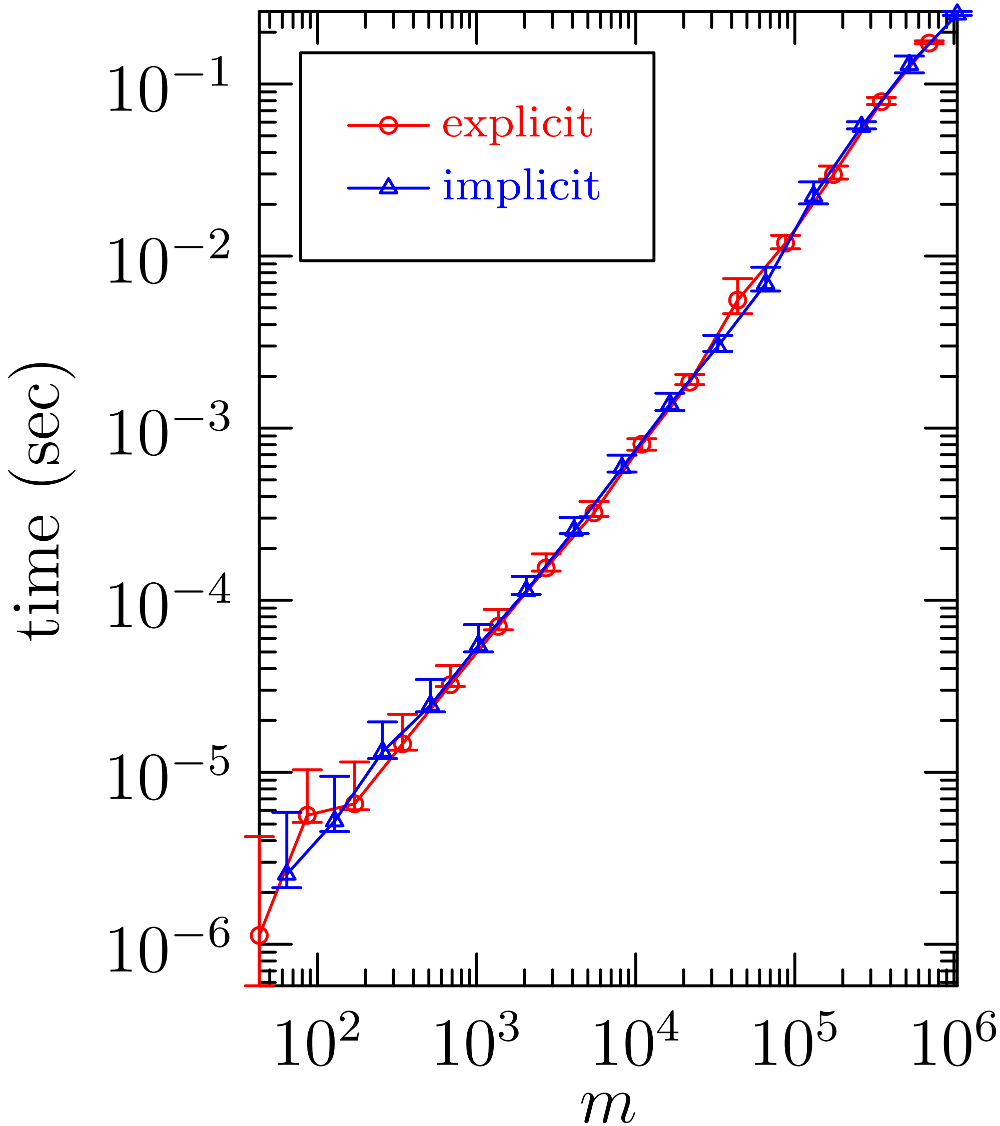
- Real-to-complex Fourier take $K\frac{N}{2}\log\frac{N}{2}$ multiplies.

- Zero-padding Hermitian data increases the array length by 50% (i.e. 2/3 padding.)

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

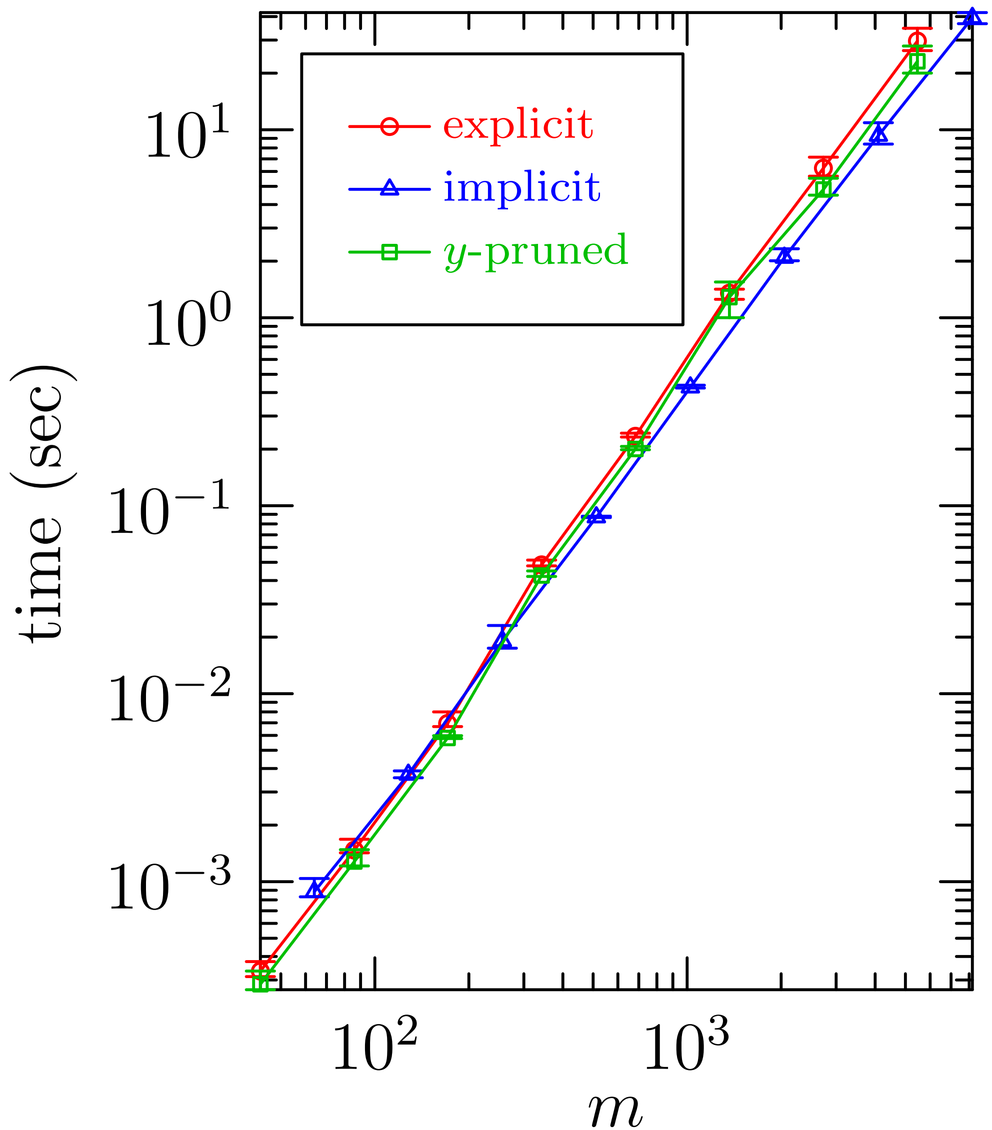
Hermitian Data

- The 1D implicit convolution is comparable to explicit padding:



Hermitian Data

- And faster in higher dimensions:



Optimal Problem Sizes

- Our main use for this algorithm is pseudo-spectral simulations.
- 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 have $N = 512$, 1024, 2048, etc.
- Phase-shift dealiasing: 2^n
  - FFTs are the same size.
- Implicit padding: $2^n - 1$.
  - sub-transforms are of size 2^{n-1}.
- Implicit padding is optimal for Mersenne-prime sized problem

Conclusion

- Implicitly padded fast convolutions eliminate aliasing errors.
- They use less memory and are faster than explicit zero-padding or phase-shift dealiasing.
- Expanding into discontiguous arrays makes for easier programming.
- A `C++` implementation under the LGPL is available at `http://fftwpp.sourceforge.net/`
- Uses SIMD routines when compiled with the Intel compiler.
- Uses the Fastest Fourier Transform in the West (`http://fftw.org/`) for sub-transforms.

References

[Canuto *et al.* 2006] C. Canuto, M. Hussaini, A. Quarteroni, & T. Zang, *Spectral Methods: Fundamentals in Single Domains*, Scientific Computation, Springer, Berlin, 2006.

[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.