The Fastest Convolution in the West

John C. Bowman and Malcolm Roberts (University of Alberta)

April 13, 2010

www.math.ualberta.ca/∼bowman/talks

Outline

- Discrete Convolutions
  - Cyclic vs. Linear
  - Standard vs. Centered
  - Complex vs. Hermitian
- Dealiasing
  - Zero Padding
  - Phase-shift dealiasing
- Implicit Padding in 1D, 2D, and 3D:
  - Standard Complex
  - Centered Hermitian
  - Biconvolution
- Conclusions

Discrete Convolutions

- Discrete linear convolution sums based on the fast Fourier transform (FFT) algorithm [Gauss 1866], [Cooley & Tukey 1965] have become important tools for:
  - image filtering;
  - digital signal processing;
  - correlation analysis;
  - pseudospectral simulations.

Discrete Cyclic Convolution

- The FFT provides an efficient tool for computing the *discrete cyclic convolution*

$$\sum_{p=0}^{N-1} F_p G_{k-p},$$

where the vectors F and G have period N.

- Define the Nth primitive root of unity:

$$\zeta_N = \exp\left(\frac{2\pi i}{N}\right).$$

- The fast Fourier transform method exploits the properties that $\zeta_N^r = \zeta_{N/r}$ and $\zeta_N^N = 1$.

- The unnormalized backwards discrete Fourier transform of $\{f_k : k = 0, \ldots, N\}$ is

$$f_j \doteq \sum_{k=0}^{N-1} \zeta_N^{jk} F_k \qquad j = 0, \ldots, N-1,$$

- The corresponding forward transform is

$$F_k \doteq \frac{1}{N} \sum_{j=0}^{N-1} \zeta_N^{-kj} f_j \qquad j = 0, \ldots, N-1.$$

- The orthogonality of this transform pair follows from

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

Discrete Linear Convolution

- The pseudospectral method requires a *linear convolution* since wavenumber space is not periodic.

- The convolution theorem states:

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

- The terms indexed by $s \neq 0$ are called *aliases*.

- We need to remove the aliases by ensuring that $G_{k-p+sN} = 0$ whenever $s \neq 0$.

- If F_p and G_{k-p+sN} are nonzero only for $0 \le p \le m-1$ and $0 \le k-p+sN \le m-1$, then we want $k+sN \le 2m-2$ to have no solutions for positive s.

- This can be achieved by choosing $N \ge 2m-1$.

- That is, one must *zero pad* input data vectors of length m to length $N \ge 2m-1$.

- Physically, *explicit zero padding* prevents mode $m-1$ from beating with itself, wrapping around to contaminate mode $N = 0 \bmod N$

- Since FFT sizes with small prime factors in practice yield the most efficient implementations, the padding is normally extended to $N = 2m$.

Pruned FFTs

- Although explicit padding seems like an obvious waste of memory and computation, the conventional wisdom on avoiding this waste is well summed up by Steven G. Johnson, coauthor of the **FFTW** ("Fastest Fourier Transform in the West") library [Frigo & Johnson]:

> *The most common case where people seem to want a pruned FFT is for zero-padded convolutions, where roughly 50% of your inputs are zero (to get a linear convolution from an FFT-based cyclic convolution). Here, a pruned FFT is hardly worth thinking about, at least in one dimension. In higher dimensions, matters change (e.g. for a 3d zero-padded array about 1/8 of your inputs are non-zero, and one can fairly easily save a factor of two or so simply by skipping 1d sub-transforms that are zero).*

Implicit Padding

- If $f_k = 0$ for $k \geq m$, one can easily avoid looping over the unwanted zero Fourier modes by decimating in wavenumber

$$f_{2\ell} = \sum_{k=0}^{m-1} \zeta_m^{\ell k} F_k, \qquad f_{2\ell+1} = \sum_{k=0}^{m-1} \zeta_m^{\ell k} \zeta_N^{k} F_k \qquad \ell = 0, 1, \ldots m-1.$$

- This requires computing two subtransforms, each of size m, for an overall computational scaling of order $2m \log_2 m = N \log_2 m$.

- Odd and even terms of the convolution can then be computed separately, multiplied term-by-term, and transformed again to Fourier space:

$$
\begin{aligned}
NF_k &= \sum_{j=0}^{N-1} \zeta_N^{-kj} f_j = \sum_{\ell=0}^{m-1} \zeta_N^{-k2\ell} f_{2\ell} + \sum_{\ell=0}^{m-1} \zeta_N^{-k(2\ell+1)} f_{2\ell+1} \\
&= \sum_{\ell=0}^{m-1} \zeta_m^{-k\ell} f_{2\ell} + \zeta_N^{-k} \sum_{\ell=0}^{m-1} \zeta_m^{-k\ell} f_{2\ell+1} \qquad k = 0, \ldots, \frac{N}{2} - 1.
\end{aligned}
$$

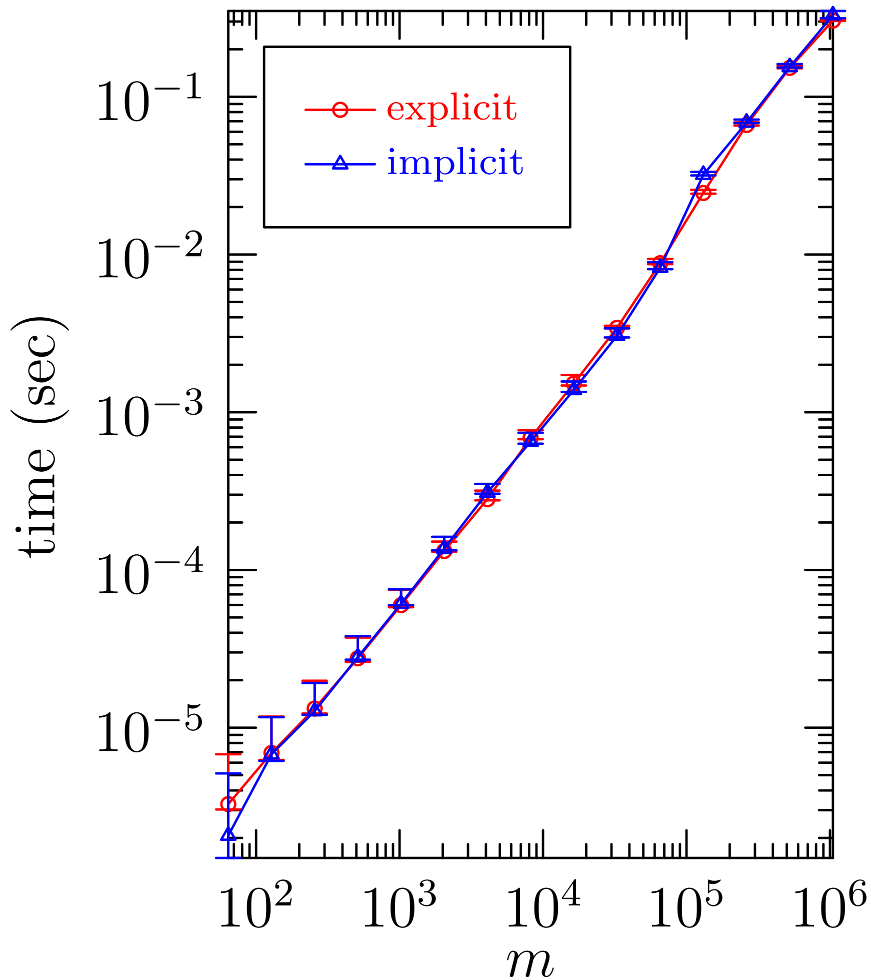
- No bit reversal is required at the highest level.

- An implicitly padded convolution is implemented as in our `FFTW++` library (version 1.05) as `cconv(f,g,u,v)` computes an in-place implicitly dealiased convolution of two complex vectors `f` and `g` using two temporary vectors `u` and `v`, each of length m.

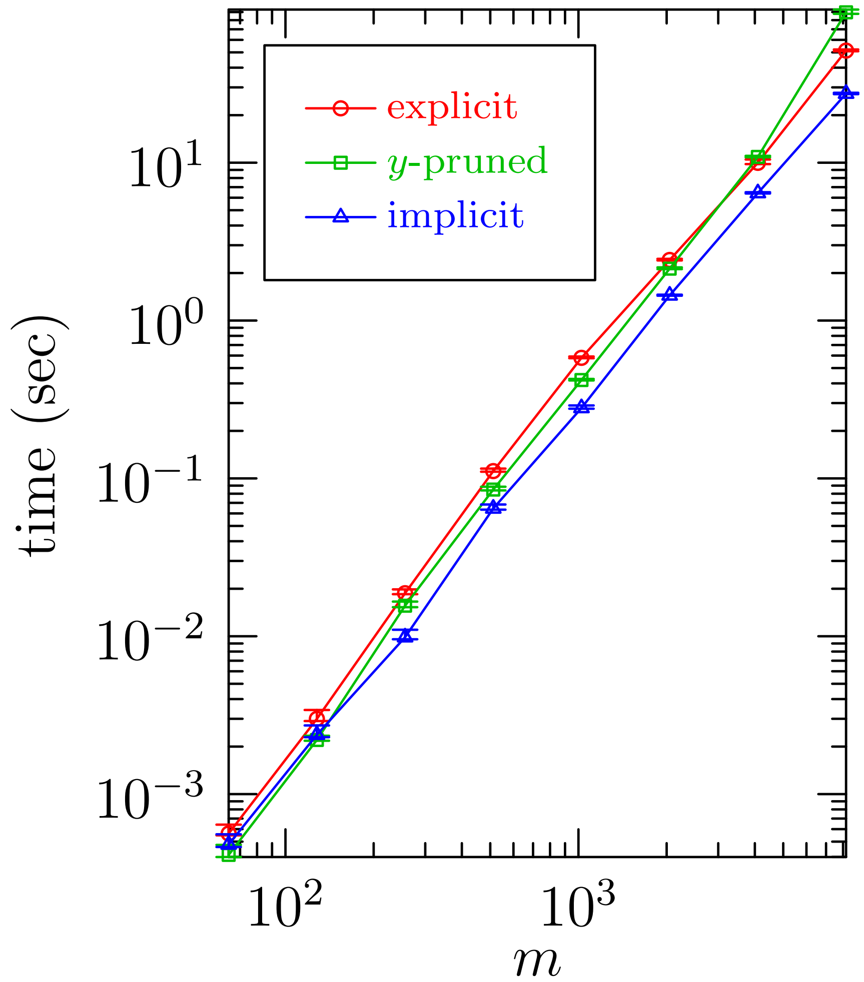
- This in-place convolution requires six out-of-place transforms, thereby avoiding bit reversal at all levels.

Input: vector f, vector g
Output: vector f
$\mathsf{u} \leftarrow \mathtt{fft}^{-1}(\mathsf{f})$;
$\mathsf{v} \leftarrow \mathtt{fft}^{-1}(\mathsf{g})$;
$\mathsf{u} \leftarrow \mathsf{u} * \mathsf{v}$;
for $k = 0$ **to** $m - 1$ **do**
  $\mathsf{f}[k] \leftarrow \zeta_{2m}^{k}\mathsf{f}[k]$;
  $\mathsf{g}[k] \leftarrow \zeta_{2m}^{k}\mathsf{g}[k]$;
end
$\mathsf{v} \leftarrow \mathtt{fft}^{-1}(\mathsf{f})$;
$\mathsf{f} \leftarrow \mathtt{fft}^{-1}(\mathsf{g})$;
$\mathsf{v} \leftarrow \mathsf{v} * \mathsf{f}$;
$\mathsf{f} \leftarrow \mathtt{fft}(\mathsf{u})$;
$\mathsf{u} \leftarrow \mathtt{fft}(\mathsf{v})$;
for $k = 0$ **to** $m - 1$ **do**
  $\mathsf{f}[k] \leftarrow \mathsf{f}[k] + \zeta_{2m}^{-k}\mathsf{u}[k]$;
end
return f/(2m);

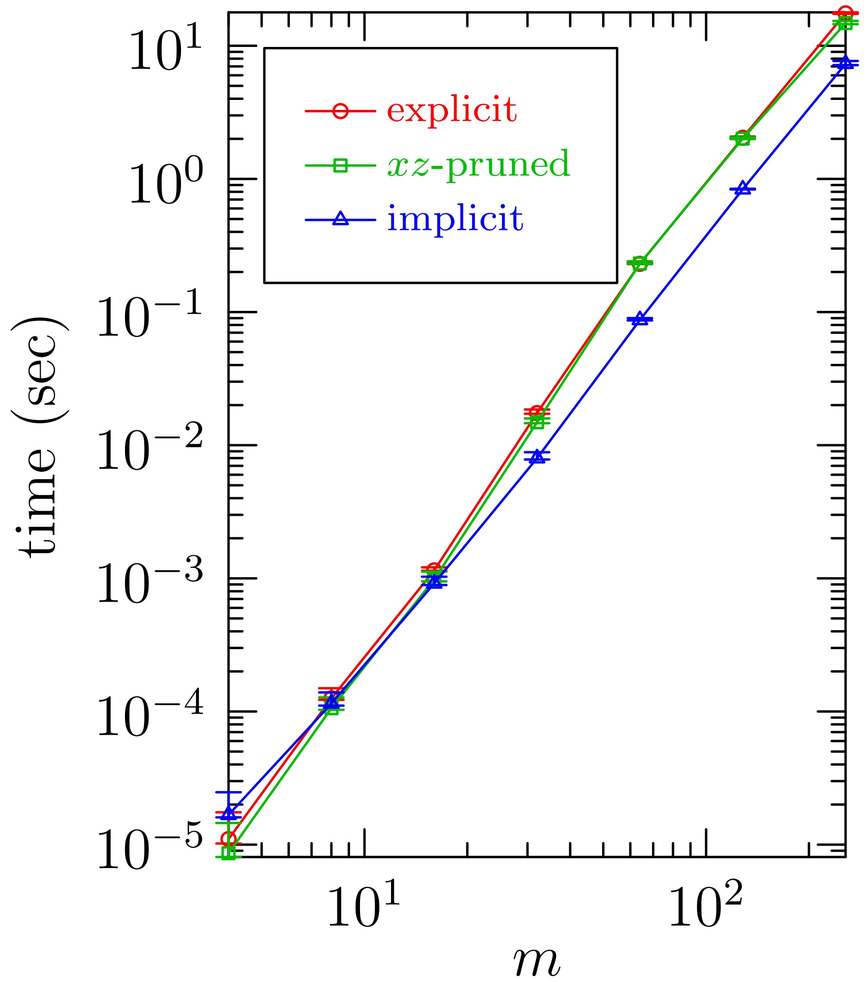
Implicit Padding in 1D



Implicit Padding in 2D



Implicit Padding in 3D



Hermitian Convolutions

- *Hermitian convolutions* arise when the input vectors are Fourier transforms of real data:

$$f_{N-k} = \overline{f_k}.$$

Centered Convolutions

- For a *centered convolution*, the Fourier origin is at wavenumber zero:

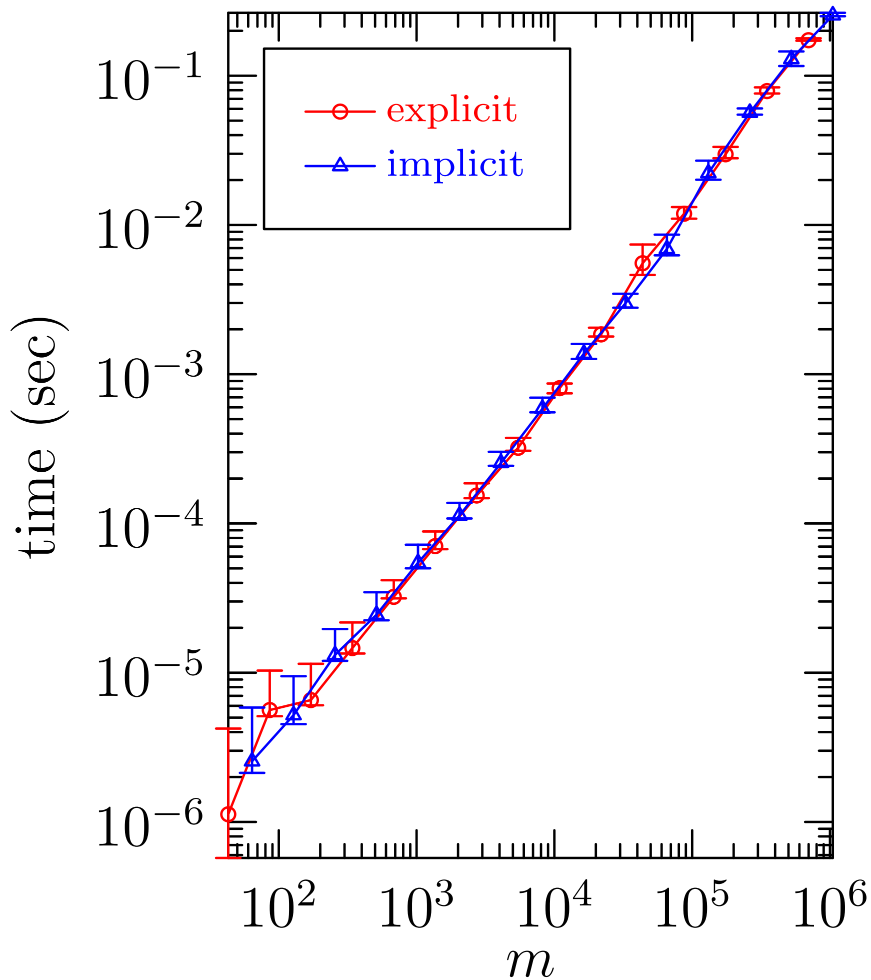
$$\sum_{p=k-m+1}^{m-1} f_p g_{k-p}$$

- Here, one needs to pad to $N \geq 3m-2$ to prevent mode $m-1$ from beating with itself to contaminate the most negative (first) mode, corresponding to wavenumber $-m+1$. Since the ratio of the number of physical to total modes, $(2m-1)/(3m-2)$ is asymptotic to $2/3$ for large m, this padding scheme is often referred to as the *2/3 padding rule.*

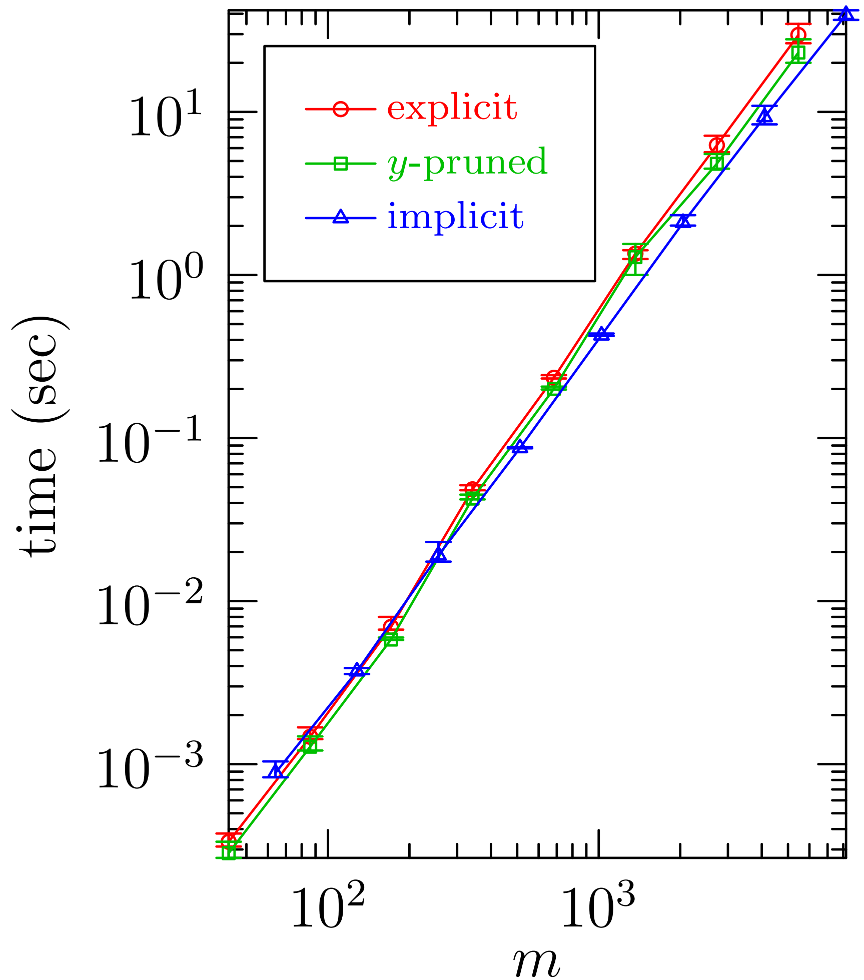
- The Hermiticity condition then appears as

$$f_{-k} = \overline{f_k}.$$

Implicit Hermitician Centered Padding in 1D



Implicit Hermitician Centered Padding in 2D



Biconvolutions

- The *biconvolution* of three vectors F, G, and H is

$$\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} F_p G_q H_{k-p-q}.$$

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

- This requires a centered Hermitian biconvolution:

$$\sum_{p=-m+1}^{m-1}\sum_{q=-m+1}^{m-1}\sum_{r=-m+1}^{m-1} F_p G_q H_r \delta_{p+q+r,k}.$$

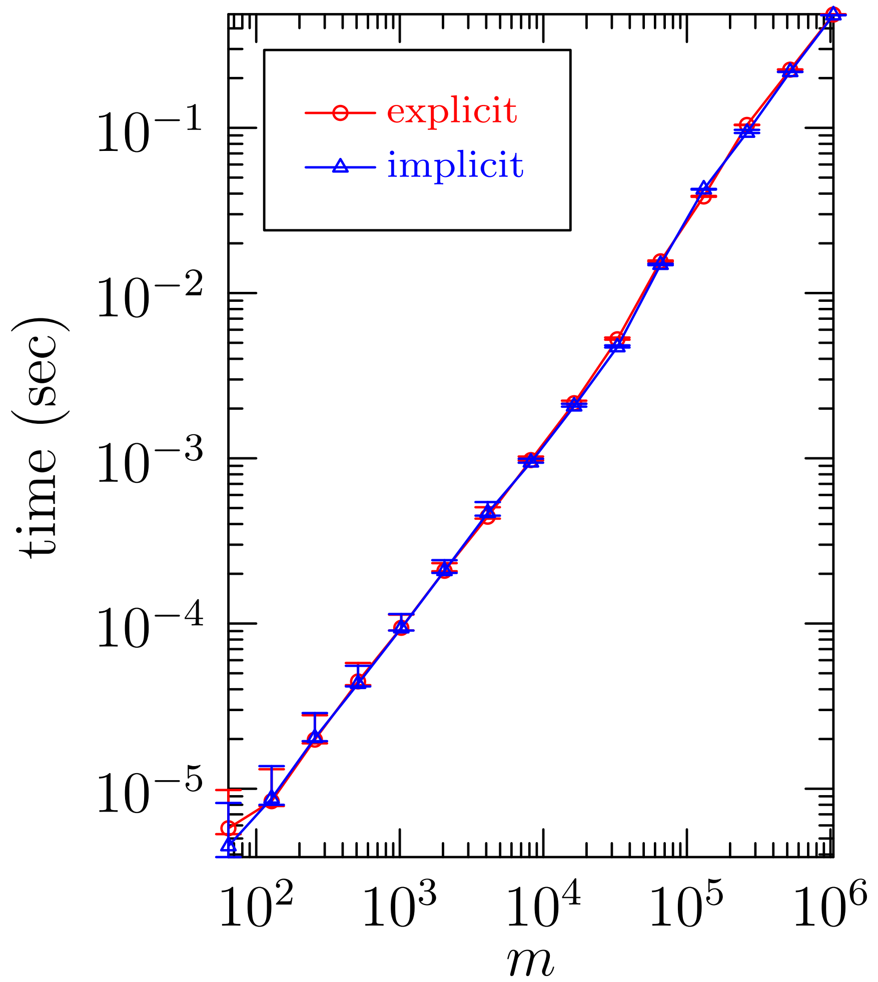
- Correctly dealiasing requires a 2/4 zero padding rule (instead of the usual 2/3 rule for a single convolution).

2/4 Padding Rule

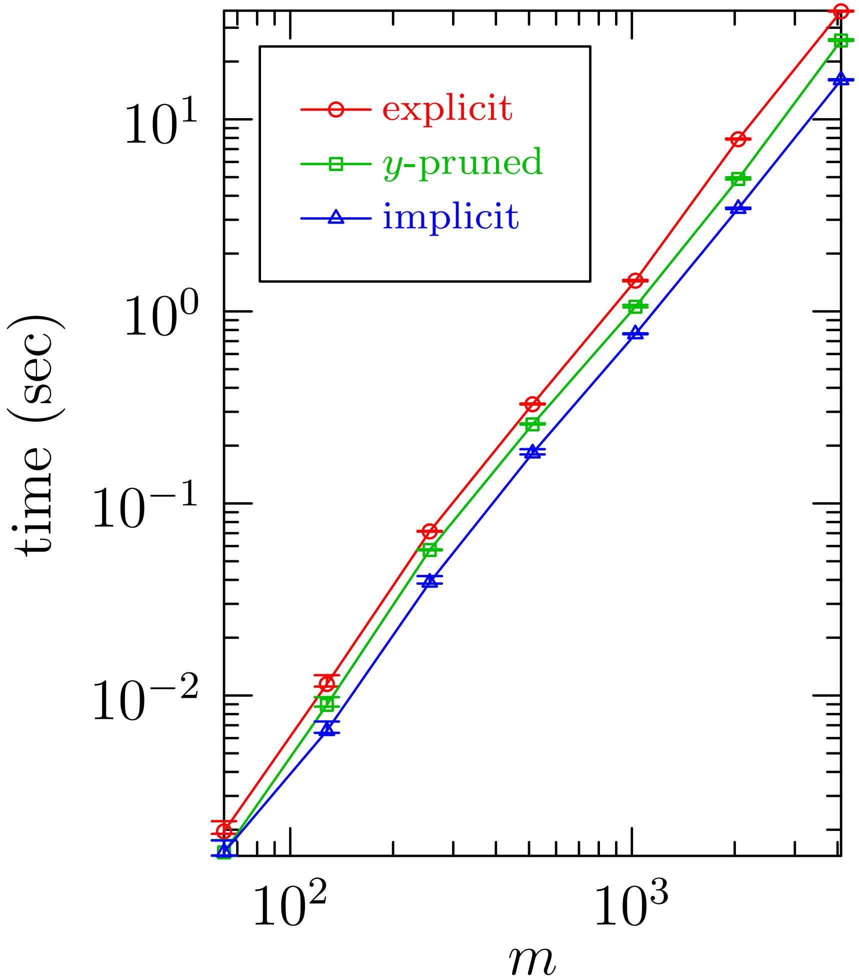
- Computing the transfer function for Z_4 with a $2/4$ padding rule means that in a 2048×2048 pseudospectral simulation, the maximum physical wavenumber retained in each direction is only 512.

- For a centered Hermitian biconvolution, implicit padding is twice as fast and uses half of the memory required by conventional explicit padding.

Implicit Biconvolution in 1D



Implicit Biconvolution in 2D



Conclusions

- Memory savings: in d dimensions implicit padding asymptotically uses $1/2^{d-1}$ of the memory require by conventional explicit padding.

- Computational savings due to increased data locality: about a factor of two.

- Highly optimized versions of these routines have been implemented as a software layer **`FFTW++`** on top of the **`FFTW`** library and released under the Lesser GNU Public License.

- With the advent of this **`FFTW++`** library, writing a high-performance dealiased pseudospectral code is now a relatively straightforward exercise.

Asymptote: 2D & 3D Vector Graphics Language

Andy Hammerlindl, John C. Bowman, Tom Prince

`http://asymptote.sf.net`

(freely available under the Lesser GNU Public License)

Asymptote Lifts TEX to 3D

$$\int_{-\infty}^{+\infty} e^{-\alpha x^2}\, dx = \sqrt{\frac{\pi}{\alpha}}$$

`http://asymptote.sf.net`

Acknowledgements: Orest Shardt (U. Alberta)

References

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

[Frigo & Johnson] "M. Frigo & S. G. Johnson, `http://www.fftw.org/pruned.html`.

[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.