# Implicit Dealiasing of Convolutions over Distributed Memory

John C. Bowman and Malcolm Roberts

University of Alberta and Aix-Marseille University

www.math.ualberta.ca/∼bowman/talks

# 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 *$N$th 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

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

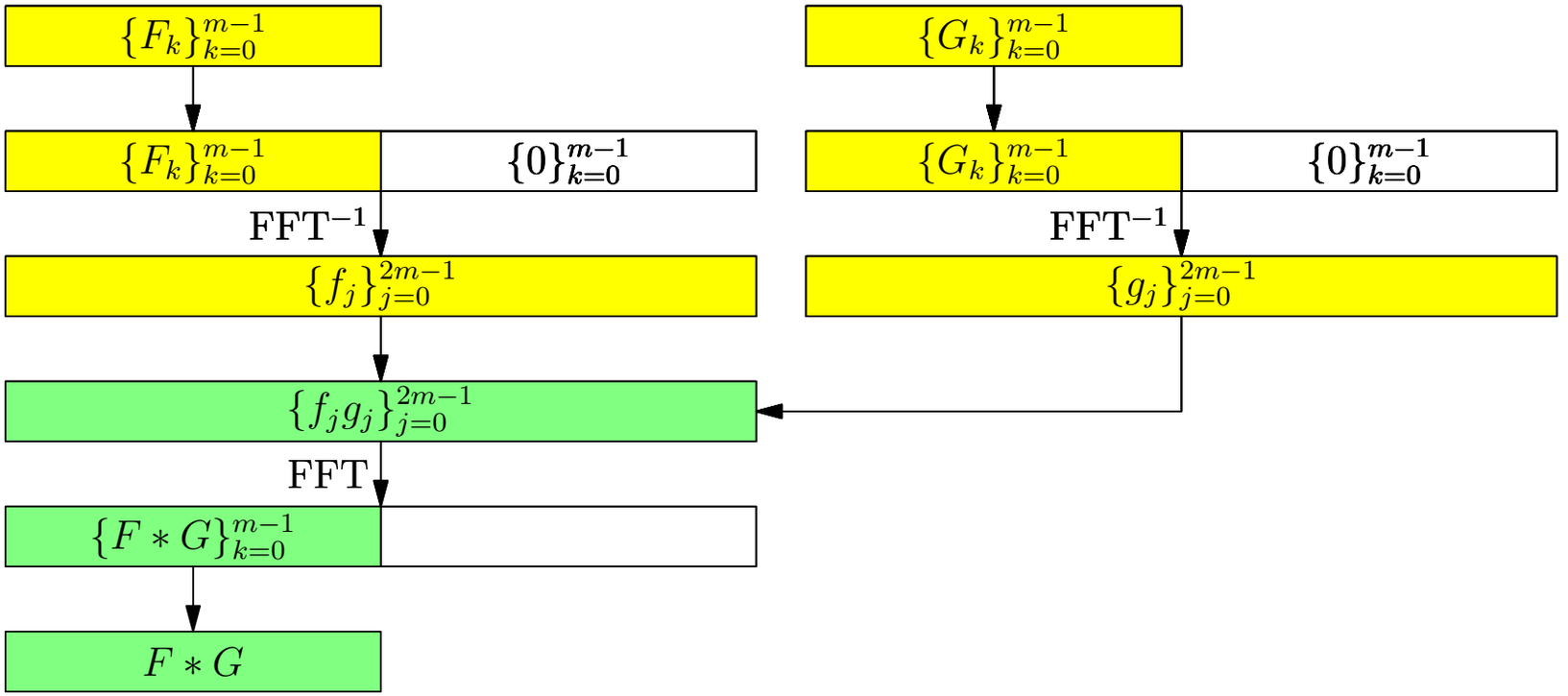
- The corresponding *forward transform is*

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

- The orthogonality of this transform pair follows from

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

- The pseudospectral method requires a *linear convolution*.
- One can dealias by *zero padding* input data vectors of length $m$ to length $N \geq 2m-1$:




- *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 :

> *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

- Let $N = 2m$. For $j = 0, \ldots, 2m-1$ we want to compute

$$f_j = \sum_{k=0}^{2m-1} \zeta_{2m}^{jk} F_k.$$

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

$$f_{2\ell} = \sum_{k=0}^{m-1} \zeta_{2m}^{2\ell k} F_k = \sum_{k=0}^{m-1} \zeta_m^{\ell k} F_k,$$

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

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

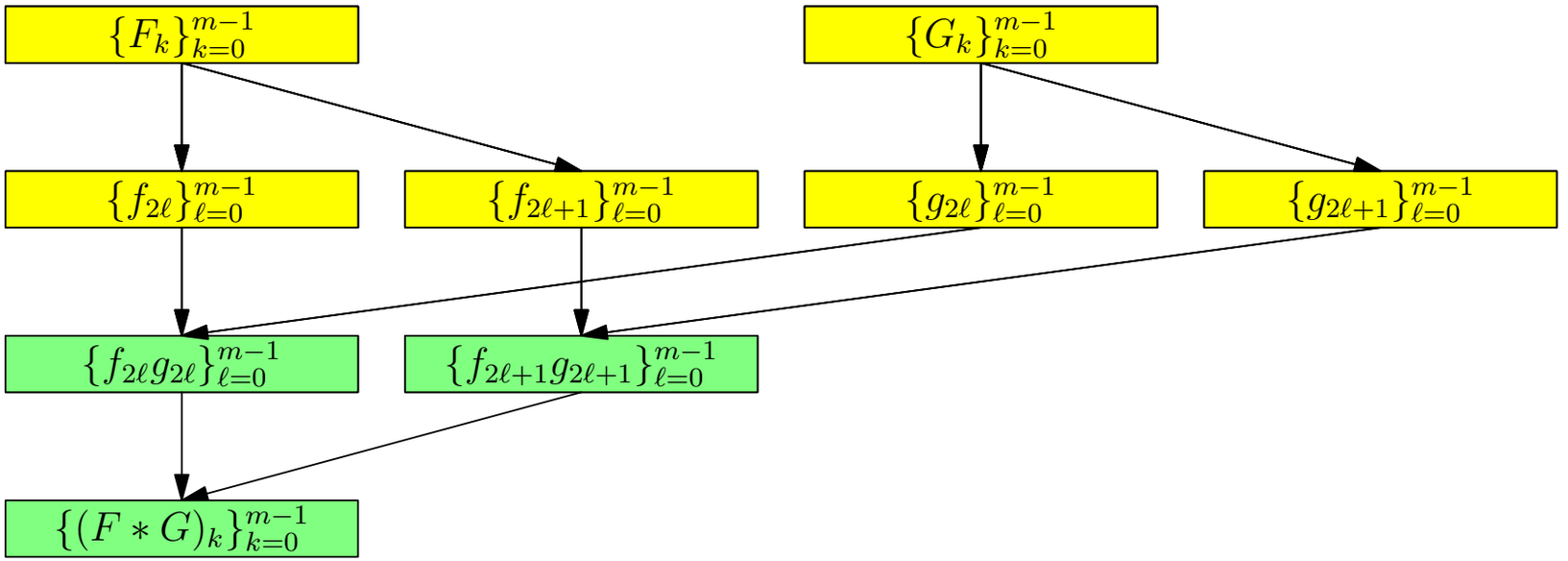
- Odd and even terms of the convolution can then be computed separately, multiplied term-by-term, and transformed again to Fourier space:

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

- No bit reversal is required at the highest level.

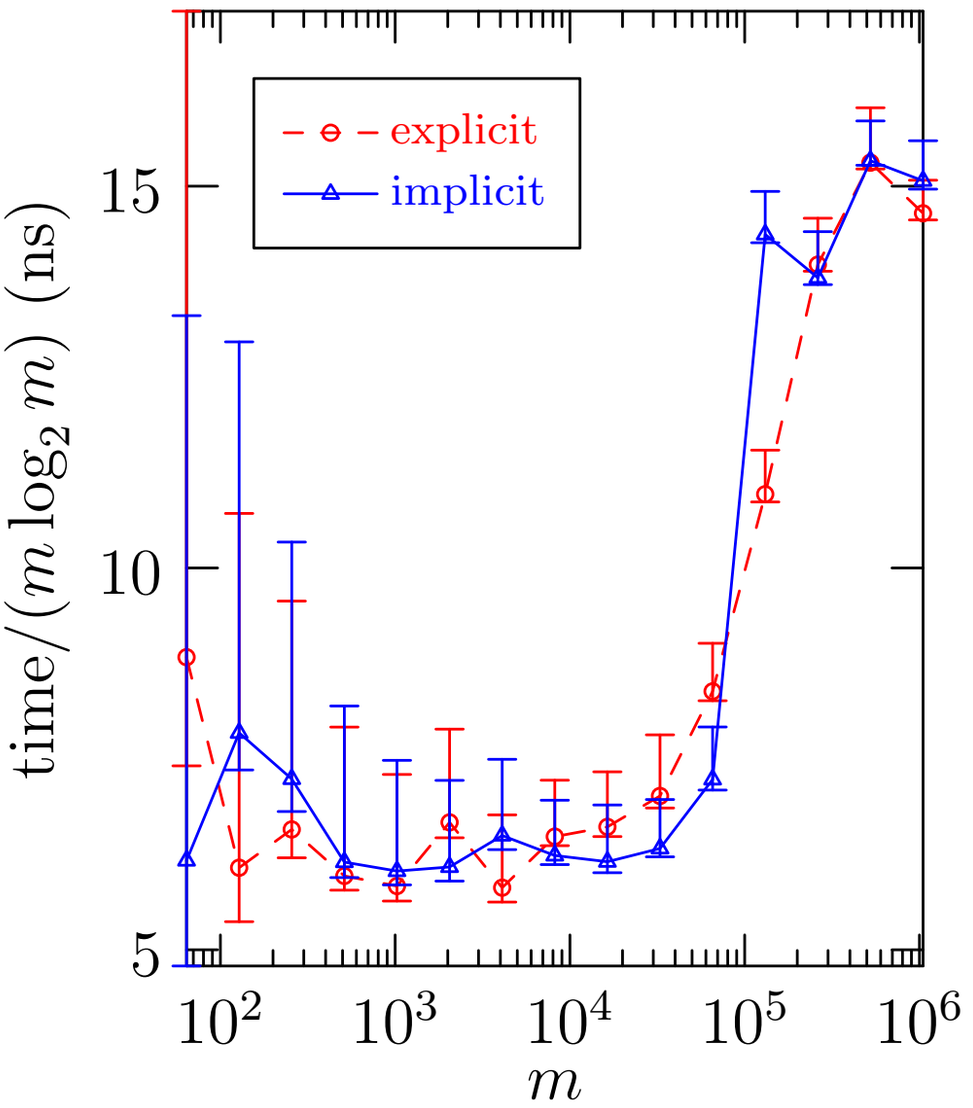
- An implicitly padded convolution is implemented as in our `FFTW++` library (version 1.13) as `cconv`(`f`,`g`,`u`,`v`) computes an in-place implicitly dealiased convolution of two complex vectors `f` and `g` using two temporary vectors `u` and `v`, each of length $m$.

- This in-place convolution requires six out-of-place transforms, thereby avoiding bit reversal at all levels.
- The computational complexity is $6Km\log_2 m$.
- The numerical error is similar to explicit padding.



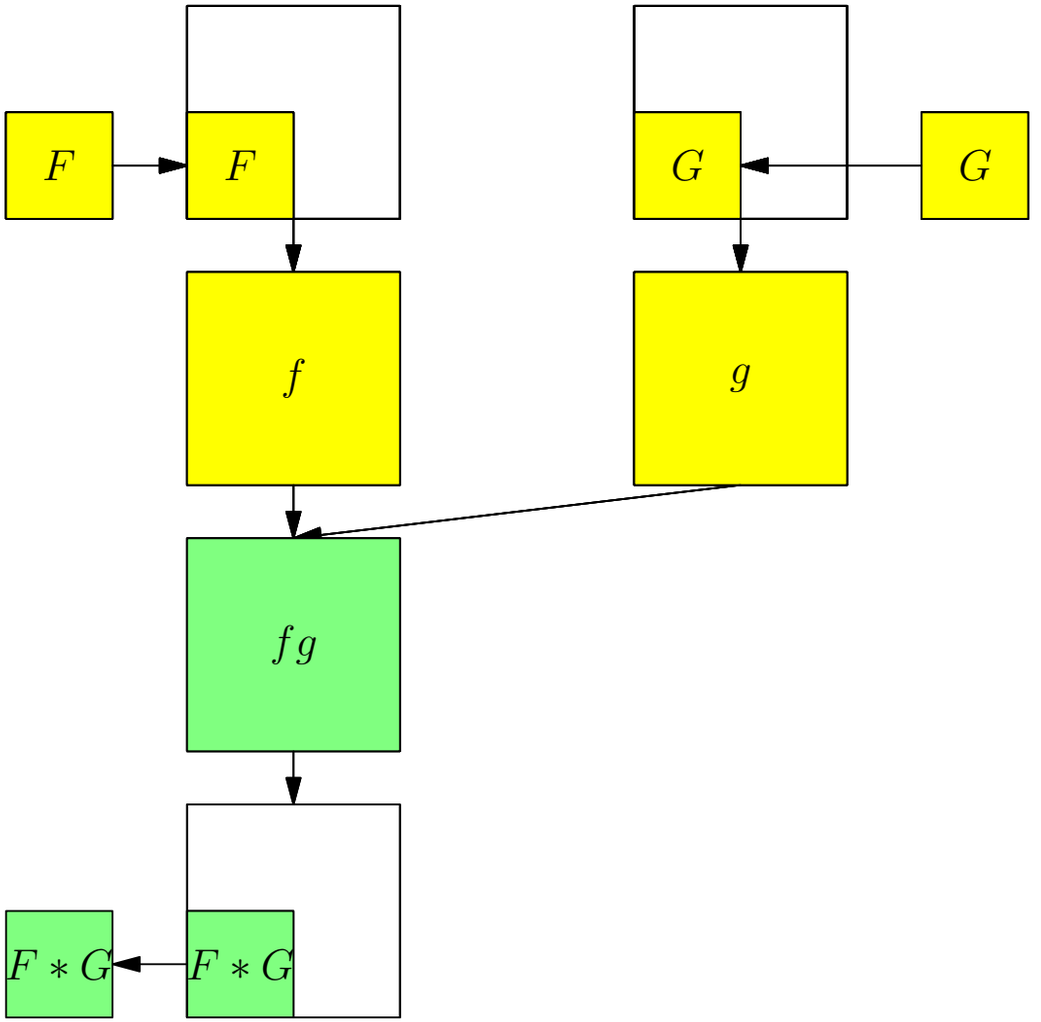
**Input**: vector $\mathsf{f}$, vector $\mathsf{g}$
**Output**: vector $\mathsf{f}$
$\mathsf{u} \leftarrow \mathtt{fft}^{-1}(\mathsf{f})$;
$\mathsf{v} \leftarrow \mathtt{fft}^{-1}(\mathsf{g})$;
$\mathsf{u} \leftarrow \mathsf{u} * \mathsf{v}$;
**for** $k = 0$ **to** $m - 1$ **do**
  $\mathsf{f}[k] \leftarrow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



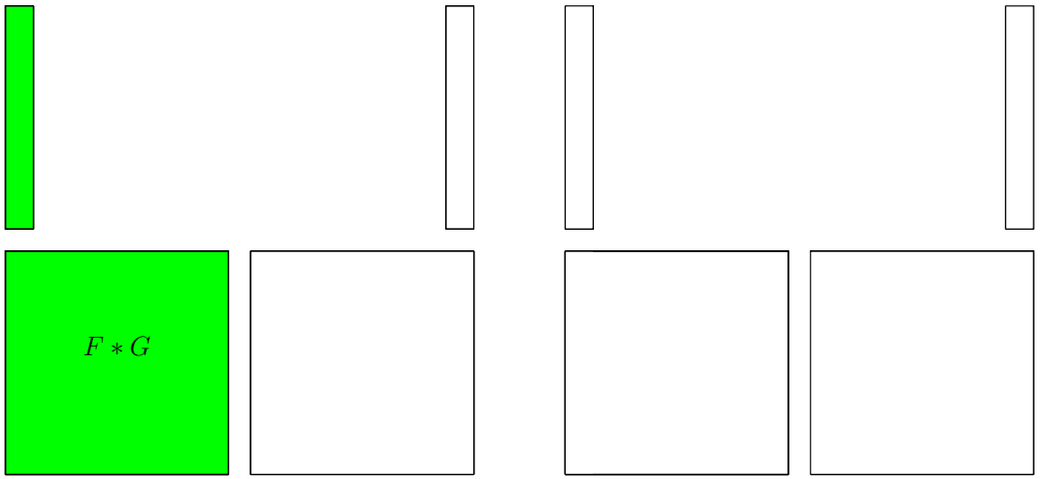
# Convolutions in Higher Dimensions

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

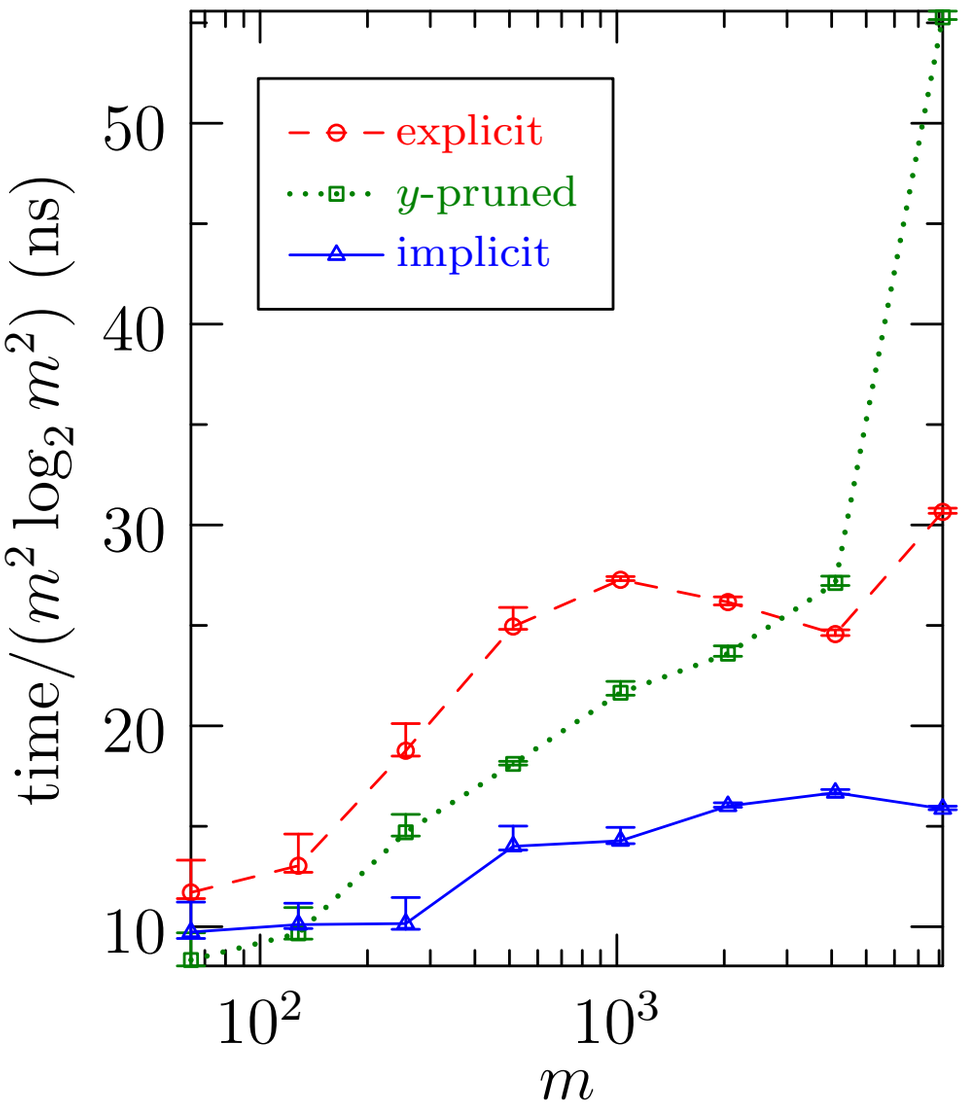


# Implicit Padding in 2D

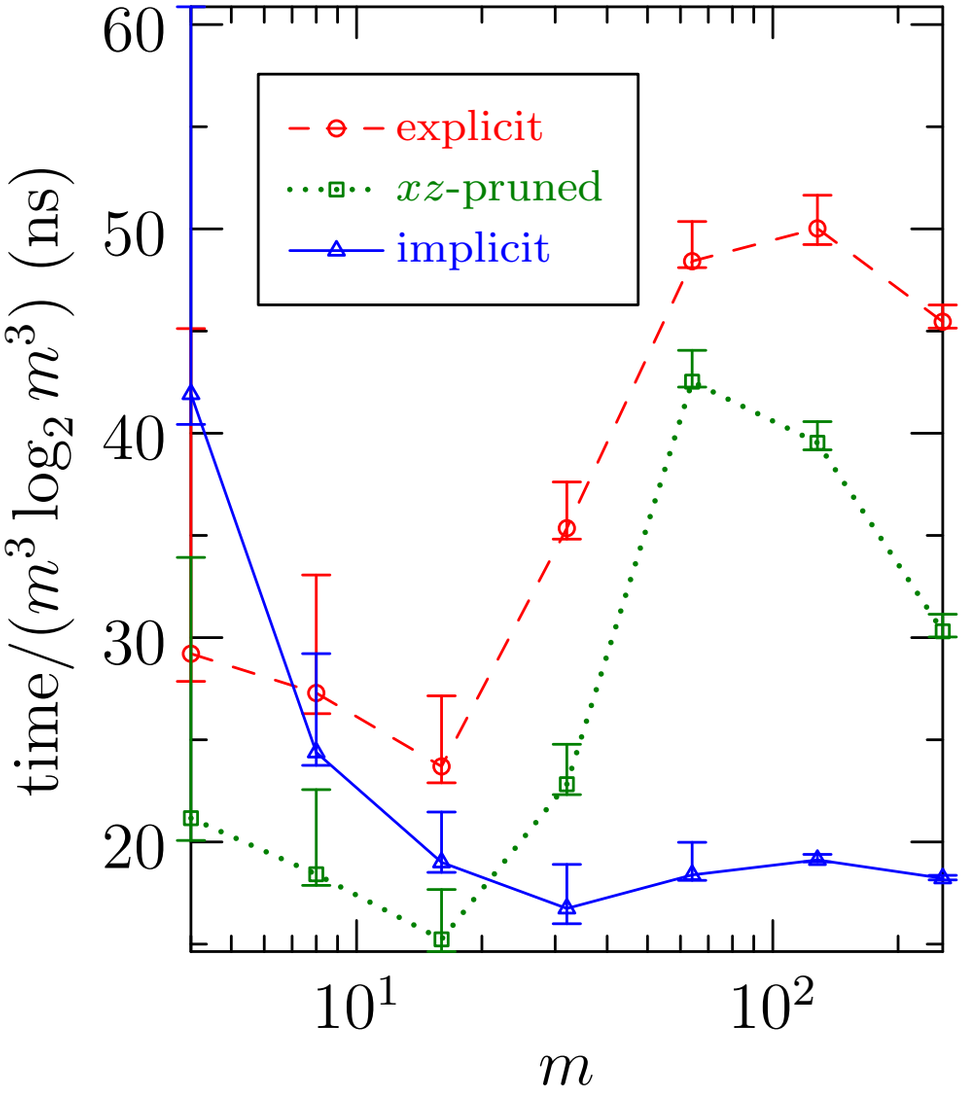
- Extra work memory need not be contiguous with the data.



# 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 ($k = 0$) is centered in the domain:

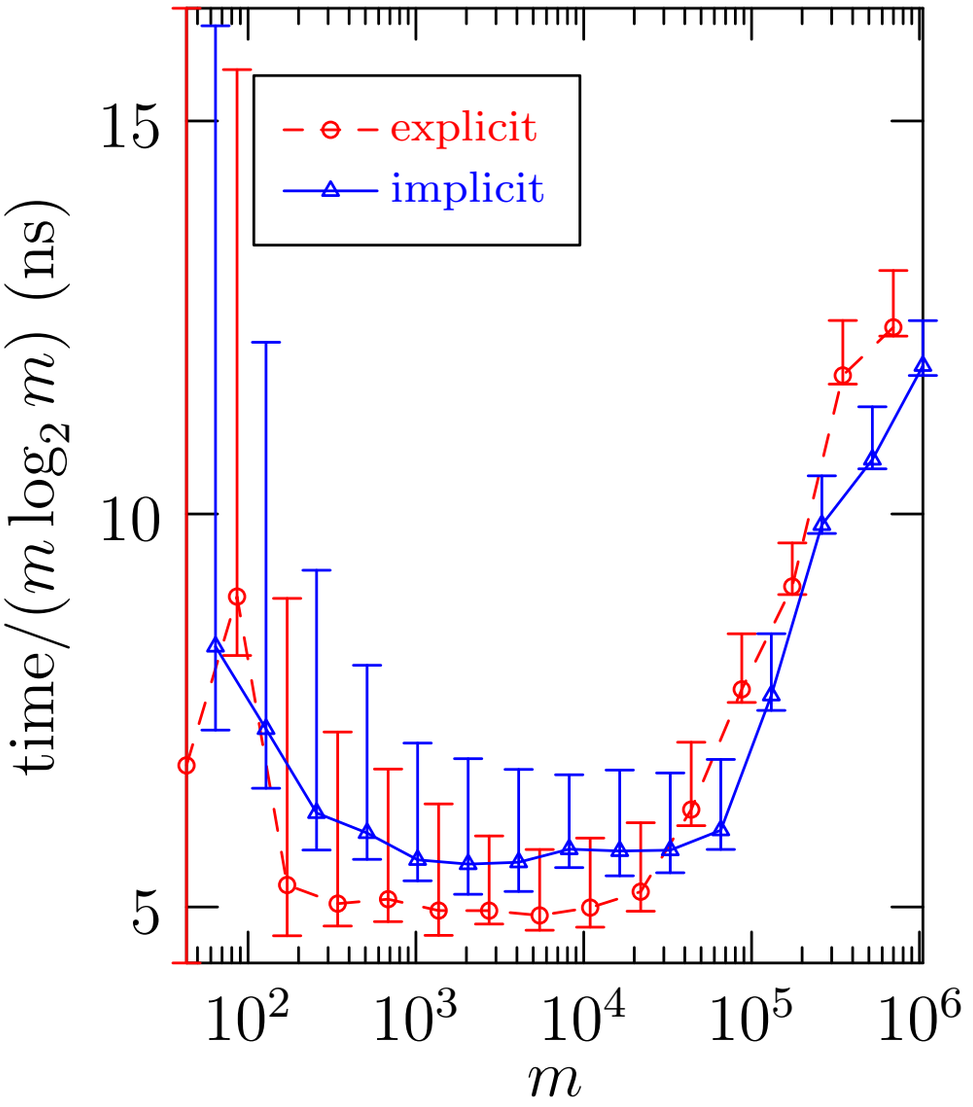
$$\sum_{p=k-m+1}^{m-1} f_p g_{k-p}$$

- 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.*

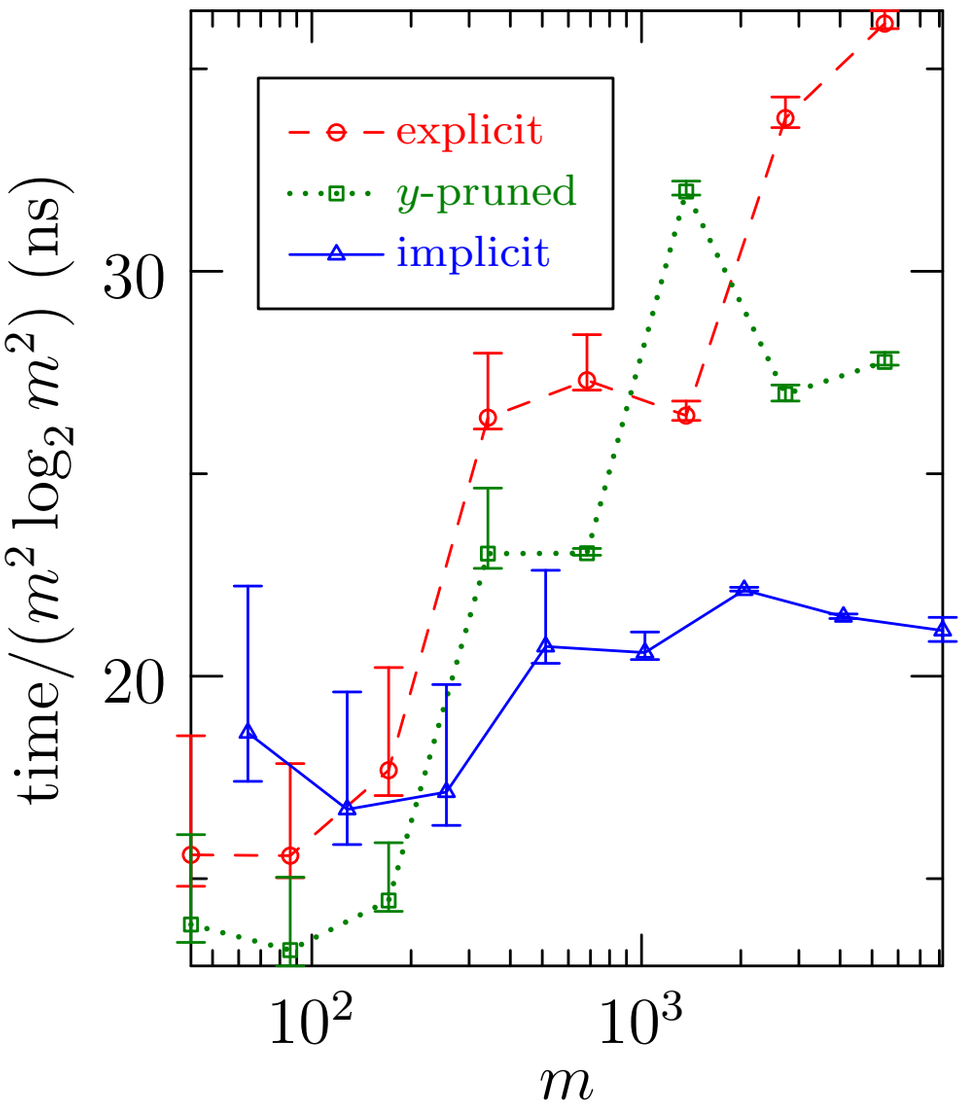
- The Hermiticity condition then appears as

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

# Implicit Hermitian Centered Padding in 1D



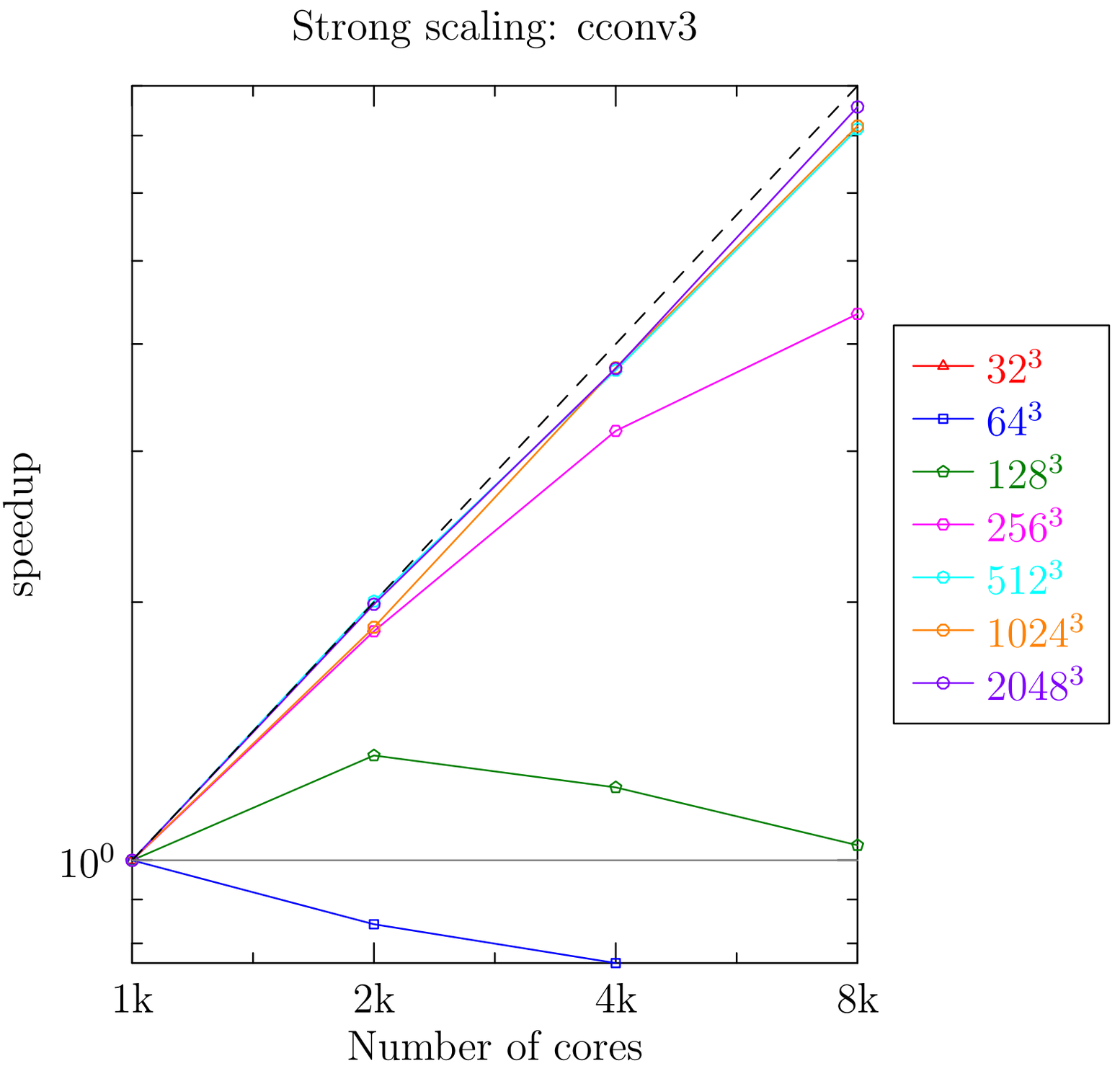
# Implicit Hermitian Centered Padding in 2D



# Parallelization

- Our implicit and explicit convolution routines have been multithreaded for shared-memory architectures.
- Parallel generalized slab/pencil model implementations have recently been developed for distributed-memory architectures (available in svn repository and upcoming 1.14 release).
- The key bottleneck is the distributed matrix transpose.
- We have compared several distributed matrix transpose algorithms, both blocking and nonblocking, under both pure MPI and hybrid MPI/OpenMP architectures.
- One advantage of hybrid MPI/OpenMP over pure MPI for matrix transposition is that it yields a larger communication block size.
- Our attempts thus far to overlap communication and computation have unfortunately resulted in fragmented communication, hurting performance.

# Pure MPI Scaling of 3D Implicit Convolution



# 2D Pseudospectral Application

- We need to compute:

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

  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

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

# 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, parallelized 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:

  `http://fftwpp.sourceforge.net/`
- Writing a high-performance dealiased pseudospectral code is now a relatively straightforward exercise!

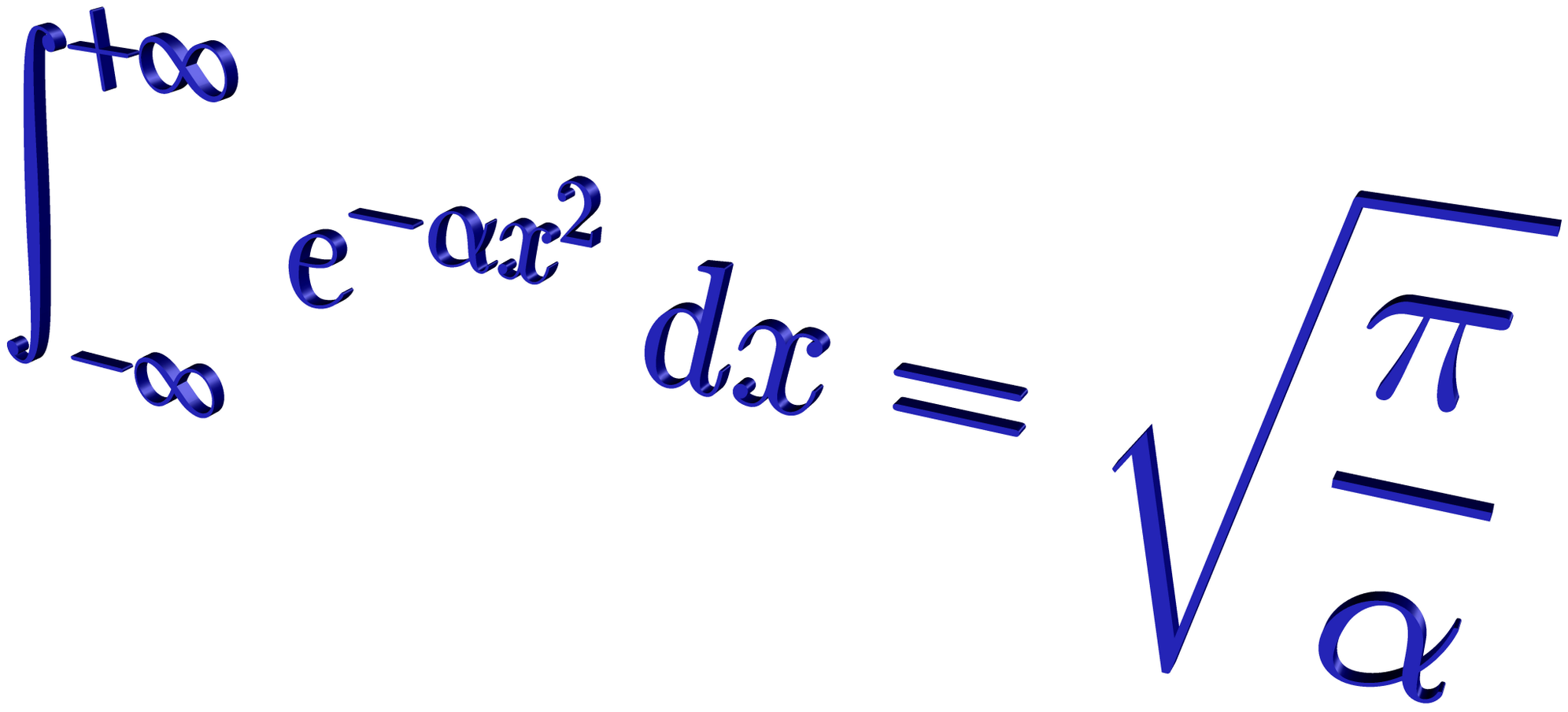
# Asymptote: 2D & 3D Vector Graphics Language


Andy Hammerlindl, John C. Bowman, Tom Prince

`http://asymptote.sf.net`

(freely available under the Lesser GNU Public License)

# Asymptote Lifts TEX to 3D




`http://asymptote.sf.net`

Acknowledgements: Orest Shardt (U. Alberta)

# References


[Bowman & Roberts 2011] J. C. Bowman & M. Roberts, SIAM J. Sci. Comput., **33**:386, 2011.

[Franchetti & Puschel 2009] F. Franchetti & M. Puschel, "Generating high performance pruned FFT implementations," in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 549–552, IEEE Computer Society, 2009.

[Frigo & Johnson ] M. Frigo & S. G. Johnson, Pruned FFTs and `FFTW`, `http://www.fftw.org/pruned.html`.

[Markel 1971] J. Markel, IEEE Transactions on Audio and Electroacoustics, **19**:305, 1971.

[Sorensen & Burrus 1993] H. Sorensen & C. Burrus, IEEE Transactions on Signal Processing, **41**:1184, 1993.