Dealiased Convolutions without the Padding

John C. Bowman and Malcolm Roberts
University of Alberta

June 20, 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
  - Ternary Convolutions

- 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 \(N\)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 \textit{backwards discrete Fourier transform} of \( \{F_k : k = 0, \ldots, N\} \) is

\[
f_j = \sum_{k=0}^{N-1} \zeta_N^{jk} F_k \quad j = 0, \ldots, N - 1.
\]

The corresponding \textit{forward transform} is

\[
F_k = \frac{1}{N} \sum_{j=0}^{N-1} \zeta_N^{-kj} f_j \quad k = 0, \ldots, N - 1.
\]

The orthogonality of this transform pair follows from

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

The pseudospectral method requires a \textit{linear convolution}. 
The Convolution Theorem

\[ \sum_{j=0}^{N-1} f_j g_j \zeta_N^{-jk} = \sum_{j=0}^{N-1} \zeta_N^{-jk} \left( \sum_{p=0}^{N-1} \zeta_N^{jp} F_p \right) \left( \sum_{q=0}^{N-1} \zeta_N^{jq} G_q \right) \]

\[ = \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} F_p G_q \sum_{j=0}^{N-1} \zeta_N^{(-k+p+q)j} \]

\[ = N \sum_{s} \sum_{p=0}^{N-1} F_p G_{k-p+sN}. \]

- The terms indexed by \( s \neq 0 \) are aliases; we need to remove them by ensuring that \( G_{k-p+sN} = 0 \) whenever \( s \neq 0 \).
- If \( F_p \) and \( G_{k-p+sN} \) are nonzero only for \( 0 \leq p \leq m - 1 \) and \( 0 \leq k - p + sN \leq m - 1 \), then we want \( k + sN \leq 2m - 2 \) to have no solutions for positive \( s \).
- This can be achieved by choosing \( N \geq 2m - 1 \).
That is, one must zero pad 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 \( N = 0 \mod N \).

- 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

- 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:
  \[
  f_{2\ell} = \sum_{k=0}^{m-1} \zeta_{2m}^{2\ell k} F_k = \sum_{k=0}^{m-1} \zeta_{m}^{\ell k} F_k,
  \]
  \[
  f_{2\ell + 1} = \sum_{k=0}^{m-1} \zeta_{2m}^{(2\ell + 1)k} F_k = \sum_{k=0}^{m-1} \zeta_{m}^{\ell k} \zeta_{N}^{k} F_k, \quad \ell = 0, 1, \ldots, m - 1.
  \]

- 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:

\[
2mF_k = \sum_{j=0}^{2m-1} \zeta_{2m}^{-kj} f_j = \sum_{\ell=0}^{m-1} \zeta_{2m}^{-k2\ell} f_{2\ell} + \sum_{\ell=0}^{m-1} \zeta_{2m}^{-k(2\ell+1)} f_{2\ell+1}
\]

\[
= \sum_{\ell=0}^{m-1} \zeta_{m}^{-k\ell} f_{2\ell} + \zeta_{2m}^{-k} \sum_{\ell=0}^{m-1} \zeta_{m}^{-k\ell} f_{2\ell+1} \quad k = 0, \ldots, m - 1.
\]

No bit reversal is required at the highest level.

An implicitly padded convolution is implemented as in our FFTW++ library (version 1.07) as \texttt{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

\[ u \leftarrow \text{fft}^{-1}(f); \]
\[ v \leftarrow \text{fft}^{-1}(g); \]
\[ u \leftarrow u \ast v; \]
\[ \text{for } k = 0 \text{ to } m - 1 \text{ do} \]
\[ f[k] \leftarrow \zeta_{2m}^k f[k]; \]
\[ g[k] \leftarrow \zeta_{2m}^k g[k]; \]
\[ \text{end} \]
\[ v \leftarrow \text{fft}^{-1}(f); \]
\[ f \leftarrow \text{fft}^{-1}(g); \]
\[ v \leftarrow v \ast f; \]
\[ f \leftarrow \text{fft}(u); \]
\[ u \leftarrow \text{fft}(v); \]
\[ \text{for } k = 0 \text{ to } m - 1 \text{ do} \]
\[ f[k] \leftarrow f[k] + \zeta_{2m}^{-k} u[k]; \]
\[ \text{end} \]
\[ \text{return } f/(2m); \]
Implicit Padding in 1D

The graph shows the relationship between time (sec) and a variable $m$. Two lines are plotted, representing Explicit and Implicit methods. The Explicit method is marked with red circles, and the Implicit method is marked with blue triangles. The y-axis is logarithmic, ranging from $10^{-1}$ to $10^{-5}$, and the x-axis is $m$, ranging from $10^2$ to $10^6$. The data points for both methods show a linear trend, with the Implicit method consistently taking longer than the Explicit method for the same value of $m$. 
Implicit Padding in 2D

![Graph showing time (sec) vs. m for explicit, y-pruned, and implicit methods.](image)
Implicit Padding in 3D

\[ \begin{align*}
10^{-1} & \quad 10^{-0} \\
10^{-2} & \quad 10^{-1} \\
10^{-3} & \quad 10^{-2} \\
10^{-4} & \quad 10^{-3} \\
10^{-5} & \quad 10^{-4} \\
\end{align*} \]

\[ \text{time (sec)} \]

\[ \text{m} \]

- explicit
- \text{\textit{xz}}-pruned
- implicit
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 Hermitician Centered Padding in 1D

![Graph showing time (sec) vs. m (m) for explicit and implicit methods. The graph indicates a linear relationship with a logarithmic scale for both axes. The explicit method is represented by red circles, and the implicit method is represented by blue triangles. The data points suggest that the implicit method takes longer than the explicit method for larger values of m.]
Implicit Hermitician Centered Padding in 2D

![Graph showing time (seconds) vs. m with different pruned and implicit methods.]
2D Pseudospectral Application

- We need to compute:

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

which appears in Fourier space as

\[
\frac{\partial \omega_k}{\partial t} = \sum_{k=p+q} \frac{p_x q_y - p_y q_x}{q^2} \omega_p \omega_q.
\]

**Input:** vector \( \omega \)

**Output:** \(- (\hat{\mathbf{z}} \times \nabla \nabla^{-2} \omega) \cdot \nabla \omega\)

**return**

ImplicitHConvolution2\((ik_x \omega, ik_y \omega, ik_y \omega / k^2, -ik_x \omega / k^2);\)
Ternary convolution

• The *ternary convolution* 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 $Z_4 = N^3 \sum_j \omega^4(x_j)$ requires computing the Fourier transform of the cubic quantity $\omega^3$.

• This requires a centered Hermitian ternary convolution:

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

• 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 ternary convolution, implicit padding is twice as fast and uses half of the memory required by conventional explicit padding.
Implicit Ternary Convolution in 1D

![Graph showing time (sec) vs m for explicit and implicit methods. The x-axis is logarithmic, ranging from $10^2$ to $10^6$, and the y-axis is also logarithmic, ranging from $10^{-5}$ to $10^{-1}$. The explicit method is represented by red circles, and the implicit method by blue triangles.]
Implicit Ternary Convolution in 2D

![Graph showing time (sec) vs. m with different convolution methods: explicit, y-pruned, and implicit.](image-url)
Conclusions

- Memory savings: in $d$ dimensions implicit padding asymptotically uses $1/2^{d-1}$ of the memory required 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 \( \text{T}\text{E}X \) to 3D

\[
\int_{-\infty}^{+\infty} e^{-ax^2} \, dx = \sqrt{\frac{\pi}{a}}
\]

http://asymptote.sf.net

Acknowledgements: Orest Shardt (U. Alberta)
References


