On the Calculation of Higher-Order Convolutions

Malcolm Roberts and John Bowman

University of Alberta

December 11, 2011

Outline

Pseudospectral simulations

Generalized convolutions

Dealiasing

Error terms

Pseudospectral simulations

- We wish to solve the Navier–Stokes equations numerically.
- The incompressible 2D vorticity formulation

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\,\omega = \nu \nabla^2 \omega$$

is Fourier-transformed into

$$\frac{\partial \omega_k}{\partial t} = \sum_{p+q=k} \frac{\epsilon_{kpq}}{q^2} \omega_p^* \omega_q^* - \nu k^2 \omega_k$$

$$\epsilon_{kpq} = (\hat{z} \cdot p \times q)\delta(k + p + q)$$

- The quadratic nonlinearity becomes a convolution.

Cubic nonlinearities

- Cubic terms
$$\frac{\partial u}{\partial t} = u^3$$
become ternary convolutions
$$\frac{\partial U_k}{\partial t} = *(U_k, U_k, U_k)$$
- The ternary convolution is defined as
$$*(F, G, H)_k = \sum_{a,b,c} F_a G_b H_c \, \delta_{k,a+b+c}.$$
- Cubic nonlinearities appear in, for example, the non-linear Schrödinger equation.

Generalized convolutions

Definition

The n-ary convolution is given by

$$*\left(F^1,\ldots,F^n\right)_k = \sum_{a_1,\ldots,a_n} F^1_{a_1}\ldots F^n_{a_n}\,\delta_{k,a_1+\cdots+a_n}.$$

Theorem

$$\boxed{\mathcal{F}^{-1}\left[*\left(F^1,\ldots,F^n\right)\right] = \prod_{i=1}^{n}\mathcal{F}^{-1}\left[F^i\right]}.$$

Generalized convolutions

Proof.

Denote roots of unity $e^{2\pi i/N} = \zeta_N$.
Let $C_k = \sum_{a_1,\dots,a_n} F^1_{a_1} \dots F^n_{a_n} \delta_{k,a_1+\dots+a_n}$.

$$\begin{aligned}
\mathcal{F}^{-1}[C]_x &= \frac{1}{N}\sum_k \zeta_N^{xk} \sum_{a_1,\dots,a_n} F^1_{a_1} \dots F^n_{a_n} \delta_{k,a_1+\dots+a_n} \\
&= \frac{1}{N}\sum_{a_1,\dots,a_n} \zeta_N^{x(a_1+\dots+a_n)} F^1_{a_1} \dots F^n_{a_n} \\
&= \prod_{i=1}^n \frac{1}{N}\sum_{a_i} \zeta_N^{xa_i} F^i_{a_i} \\
&= \prod_{i=1}^n \mathcal{F}^{-1}\left[F^i\right]_x.
\end{aligned}$$

□

Dealiasing n-ary convolutions

- Direct convolutions require $\mathcal{O}(N^2)$ operations.
- FFT-based convolutions require $\mathcal{O}(N \log N)$ operations.
- However, FFTs treat input data as periodic: this produces aliasing errors.
- We remove these by implicitly padding the data with zeroes.

Dealiasing n-ary convolutions

For non-centered input data,

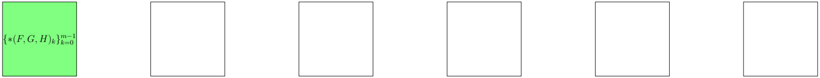
- $F^i = \{F^i_k, k \in (0, \dots, m-1)^d\}$,
- Pad from $(0, \dots, m-1)^d$ to length $(0, \dots, nm-1)^d$.
- Explicit padding memory use: $n^{d+1} m^d$,
- Implicit padding memory use: $n^2 m^d$,
- Complexity: $n(n+1)\frac{n^d-1}{n-1} K m^d \log nm$.

Dealiasing n-ary convolutions

For Hermitian-symmetric centered input data,

- $F^i = \{F^i_k, k \in (-m+1, \dots, m-1)^d\}$,
- $F_{-k} = F^*_k$, where the asterisk is complex conjugation,
- Pad from $\{0, m-1\} \times \{-m+1, m-1\}^{d-1}$ to $\{0, nm-1\} \times \{-m+1, nm-1\}^{d-1}$.
- Explicit padding memory use: $\frac{n(n+1)^d}{2} m^d$,
- Implicitly padding memory use: $n(n+1)2^{d-2}m^d$,
- Complexity: $\frac{1}{2}(n+1)^2 \frac{(n+1)^d - 2^d}{n-1} Km^d \log nm$.

Exposition of implicitly padded $2D$ ternary convolutions



Calculating using binary convolutions

n-ary convolutions are very expensive.

- Can we use binary convolutions?
- For periodic data, non-centered data, and infinite-length input data,

$$\boxed{*(F, G, H) = (F * G) * H.}$$

For n-ary non-centered convolutions, using binary convolutions is better:

- requires only $(n+2)m^d$ memory,
- has complexity $(n-1)6(2^d-1)\,Km^d \log m$.

Calculating using binary convolutions: proof

Theorem

$*(F^1, \dots, F^n)$, $F^i \in \ell^2$ can be computed as binary convolutions.

Proof.

$$
\begin{aligned}
&*\left(F^1, \dots, F^n\right)_k \\
&= \sum_{a_1,\dots,a_n \in \mathbb{Z}} F^1_{a_1} \dots F^n_{a_n} \delta_{a_1+\dots+a_n,k} \\
&= \sum_{a_1,\dots,a_n \in \mathbb{Z}} F^1_{a_1} \dots F^n_{a_n} \delta_{a_1+\dots+a_{n-1},\widehat{k}}\, \delta_{a_n+\widehat{k},k} \\
&= \sum_{\widehat{k},a_n \in \mathbb{Z}} F^n_{a_n} \delta_{a_n+\widehat{k},k} \sum_{a_1,\dots,a_{n-1} \in \mathbb{Z}} F^1_{a_1} \dots F^{n-1}_{a_{n-1}} \delta_{a_1+\dots+a_{n-1},\widehat{k}} \\
&= \left[*\left(F^1, \dots, F^{n-1}\right) * F^n\right]_k.
\end{aligned}
$$

□

Centered data

Theorem

For centered data, $\boxed{*(F, G, H) \neq (F * G) * H}$.

Proof.

$*(F_a, G_b, H_c)_1$			$(F_a * (G_b * H_c)_{\widehat{k}})_1$			
a	b	c	a	$\widehat{k}$	b	c
1	0	0	1	0	0	0
0	1	0	0	1	1	0
0	0	1	0	1	0	1
1	1	-1	1	0	1	-1
1	-1	1	1	0	-1	1
-1	1	1	N/A			

Moreover, $F * (G * H) \neq (F * G) * H$. □

Centered data

We can reduce the memory use by multiplying arrays in pairs.

For 1D convolutions,

- memory use is reduced to $\frac{3}{2}nm$,
- computational complexity is unchanged.
- Not applicable to multi-dimensional transforms.

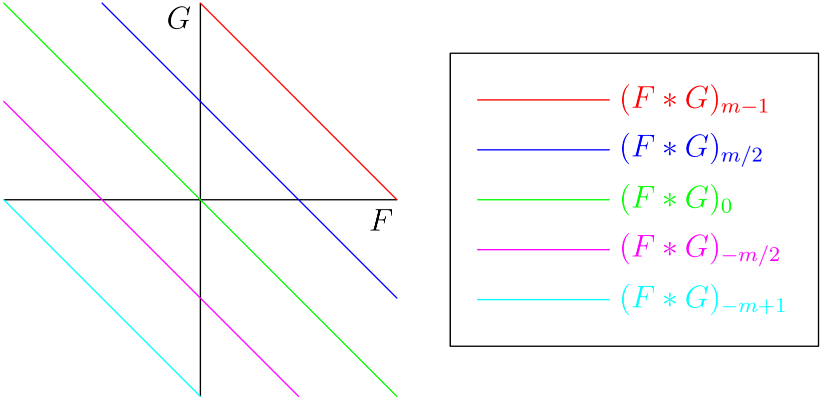

$\{f_n\}_{n=0}^{N-1}$
$\{g_n\}_{n=0}^{N-1}$
$\{h_n\}_{n=0}^{N-1}$
$\{f_n\}_{n=0}^{N-1}$
$\{F_k\}_{k=0}^{2N-1}, k$ odd
$\{g_n\}_{n=0}^{N-1}$
$\{G_k\}_{k=0}^{2N-1}, k$ odd
$\{h_n\}_{n=0}^{N-1}$
$\{f_n\}_{n=0}^{N-1}$
$\{F_k G_k\}_{k=0}^{2N-1}, k$ odd
$\{g_n\}_{n=0}^{N-1}$
$\{h_n\}_{n=0}^{N-1}$
$\{f_n\}_{n=0}^{N-1}$
$\{F_k G_k\}_{k=0}^{2N-1}, k$ odd
$\{g_n\}_{n=0}^{N-1}$
$\{H_k\}_{n=0}^{N-1}, k$ odd
$\{h_n\}_{n=0}^{N-1}$
$\{f_n\}_{n=0}^{N-1}$
$\{g_n\}_{n=0}^{N-1}$
$\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{h_n\}_{n=0}^{N-1}$
$\{f_n\}_{n=0}^{N-1}$
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{g_n\}_{n=0}^{N-1}$
$\{h_n\}_{n=0}^{N-1}$
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{g_n\}_{n=0}^{N-1}$
$\{F_k\}_{k=0}^{2N-1}, k$ even
$\{h_n\}_{n=0}^{N-1}$
$\{G_k\}_{k=0}^{2N-1}, k$ even
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{F_k\}_{k=0}^{2N-1}, k$ even
$\{h_n\}_{n=0}^{N-1}$
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{F_k G_k\}_{k=0}^{2N-1}, k$ even
$\{h_n\}_{n=0}^{N-1}$
$\{H_k\}_{k=0}^{2N-1}, k$ even
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{F_k G_k\}_{k=0}^{2N-1}, k$ even
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ even
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ even
$\mathcal{F}^{-1}\{F_k G_k H_k\}_{k=0}^{2N-1}, k$ odd
$\{*(f,g,h)\}_{n=0}^{2N-1}$

Error terms

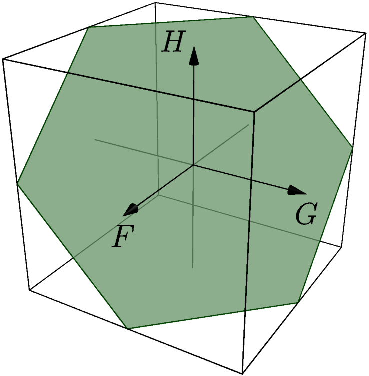
- The full n-ary convolution is very expensive.
- Using binary convolutions to calculate an n-ary convolution is cheap, but misses terms.
- The missed terms are far from the origin.
- If the input data decreases with $|k|$, binary convolutions might produce reasonable results.

Error terms

To visualize which terms are missing in $F * (G * H)$, first consider the convolution of F and G:

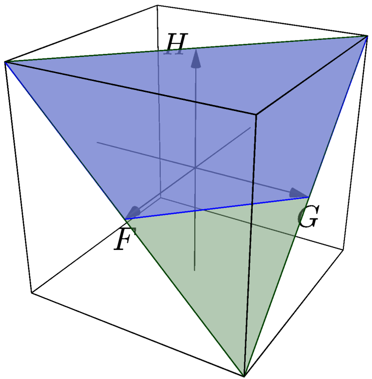


Error terms



$*(F, G, H)_0$ and $((F * G) * H)_0$ agree.

Error terms



$*(F, G, H)_{m-1}$ and $((F * G) * H)_{m-1}$ disagree.

Conclusions

- Pseudospectral simulations with higher nonlinearities involve computing n-ary convolutions.
- Non-centered n-ary convolutions can be computed via binary convolutions.
- Centered n-ary convolutions cannot be computed via binary convolutions.
- The difference error occurs at high wavenumbers; it may be possible to bound the error.

Future Work

- Parallelize the implicit convolution.
- Optimize parallel routines for Navier–Stokes simulators.
- Determine if reduced-memory n-ary convolutions are worthwhile for large values of n.
- Develop code for special cases, such as self-convolution.

Resources

FFTW++:
http://fftwpp.sourceforge.net

Asymptote:
http://asymptote.sourceforge.net

Malcolm Roberts:
http://www.math.ualberta.ca/~mroberts