The Partial Fourier Transform

John C. Bowman and Zayd Ghoggali
Department of Mathematical and Statistical Sciences
University of Alberta

July 19, 2017

www.math.ualberta.ca/∼bowman/talks

Partial Fourier Transform

- The backward 1D discrete partial Fourier transform of a complex vector $\{F_k : k = 0, \ldots, N-1\}$ is defined as

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

where $\zeta_N = e^{2\pi i/N}$ denotes the Nth primitive root of unity.

- The partial Fourier transform has applications in seismology.
- It can also be used to decompose turbulent inertial-range transfers into nonlocal and local contributions.
- The special case $c(j) = N - 1$ reduces to the usual 1D DFT:

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

DFT Conventions

- The *backward discrete Fourier transform* of $\{F_k : k = 0, \ldots, N-1\}$ 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

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

Fractional-Phase Fourier Transform

- Consider the partial fractional-phase Fourier transform:

$$f_j \doteq \sum_{k=0}^{c(j)} \zeta^{\alpha jk} F_k, \qquad j = 0, \ldots, N-1,$$

where $\zeta \doteq \zeta_1 = e^{2\pi i}$ and $\alpha \in \mathbb{R}$.

Special Case of Partial DFT: $c(j) = j$

- Given inputs $\{F_k : k = 0, \ldots, N-1\}$,

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

- Use $jk = \frac{1}{2}\left[j^2 + k^2 - (j-k)^2\right]$: [Bluestein 1970]

$$f_j = \sum_{k=0}^{j} \zeta^{\frac{\alpha}{2}\left[j^2+k^2-(j-k)^2\right]} F_k = \zeta^{\alpha j^2/2} \sum_{k=0}^{j} \zeta^{\alpha k^2/2} F_k \zeta^{-\alpha (j-k)^2/2}.$$

- This can be written as the convolution of the two sequences $g_j \doteq \zeta^{\alpha j^2/2}$ and $h_k \doteq g_k F_k$:

$$f_j = g_j \sum_{k=0}^{j} h_k \overline{g}_{j-k}.$$

Evaluating the Partial Convolution

- We wish to compute the (partial) convolution

$$\sum_{k=0}^{j} h_k \overline{g}_{j-k}.$$

- Prepend N zeros to the sequences $\{g_k\}$ and $\{h_k\}$, indexed as $k = -N, -N+1, \ldots, -1$, so that

$$\sum_{k=0}^{j} h_k \overline{g}_{j-k} = \sum_{k=-N}^{N-1} h_k \overline{g}_{j-k}.$$

- The added zeros also avoid aliases when using the cyclic DFT to compute a linear convolution.

- The convolution can then be efficiently computed using a cyclic discrete Fourier transform of length $2N$:

$$
\begin{aligned}
\sum_{k=-N}^{N-1} h_k \overline{g}_{j-k} &= \frac{1}{(2N)^2} \sum_{k=-N}^{N-1} \sum_{\ell=0}^{2N-1} \zeta_{2N}^{-k\ell} H_\ell \sum_{m=0}^{2N-1} \zeta_{2N}^{-(j-k)m} G_m \\
&= \frac{1}{(2N)^2} \sum_{\ell=0}^{2N-1} \sum_{m=0}^{2N-1} \zeta_{2N}^{-jm} H_\ell G_m \sum_{k=-N}^{N-1} \zeta_{2N}^{k(m-\ell)} \\
&= \frac{1}{(2N)^2} \sum_{\ell=0}^{2N-1} \sum_{m=0}^{2N-1} \zeta_{2N}^{-jm} H_\ell G_m \, 2N \delta_{\ell m} \\
&= \frac{1}{2N} \sum_{\ell=0}^{2N-1} \zeta_{2N}^{-j\ell} H_\ell G_\ell,
\end{aligned}
$$

where $H_\ell = \sum_{k=-N}^{N-1} \zeta_{2N}^{\ell k} h_k$ and $G_m = \sum_{k=-N}^{N-1} \zeta_{2N}^{mk} \overline{g}_k$ are the discrete Fourier transforms of $\{h_k\}$ and $\{\overline{g}_k\}$.

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.

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

Convolution Theorem

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

- The terms indexed by $s \neq 0$ are *aliases;* we need to remove them!
- If only the first m entries of the input vectors are nonzero, aliases can be avoided 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 $N = 0 \bmod N$.

Implicit Dealiasing

- 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:

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

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

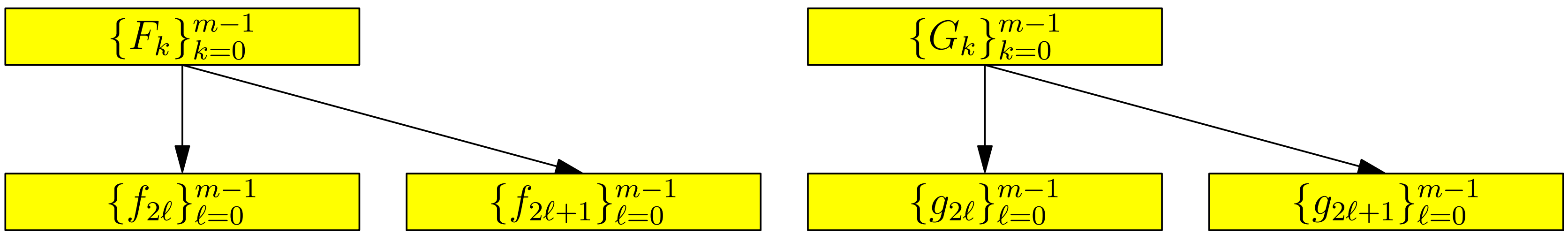
- Parallelized multidimensional implicit dealiasing routines have been implemented as a software layer **FFTW++** (v 2.05) on top of the **FFTW** library under the Lesser GNU Public License:

`http://fftwpp.sourceforge.net/`

$\{F_k\}_{k=0}^{m-1}$ $\{G_k\}_{k=0}^{m-1}$

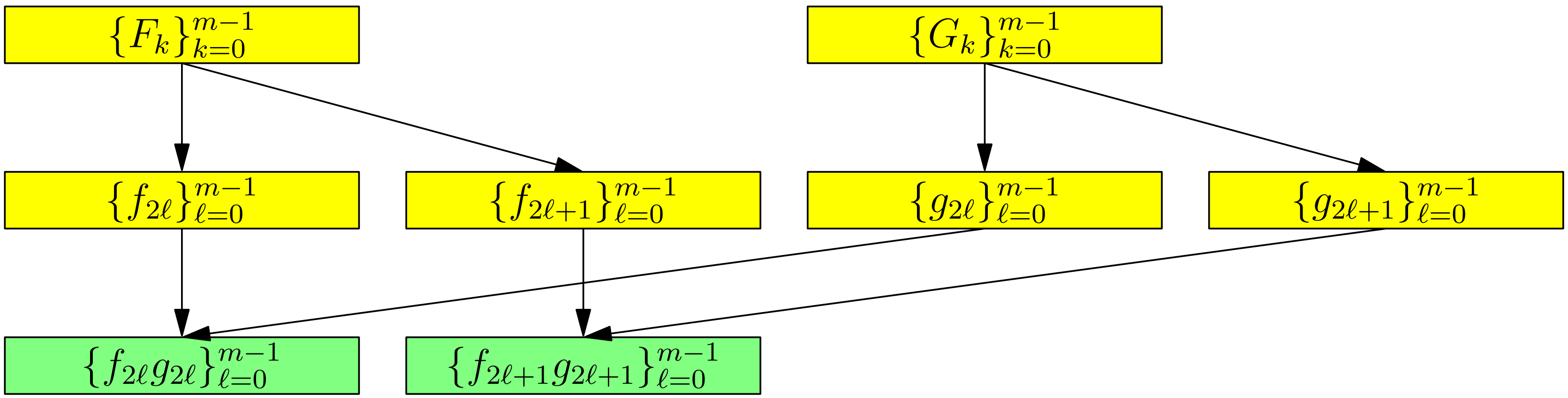
- Parallelized multidimensional implicit dealiasing routines have been implemented as a software layer **FFTW++** (v 2.05) on top of the **FFTW** library under the Lesser GNU Public License:

  `http://fftwpp.sourceforge.net/`



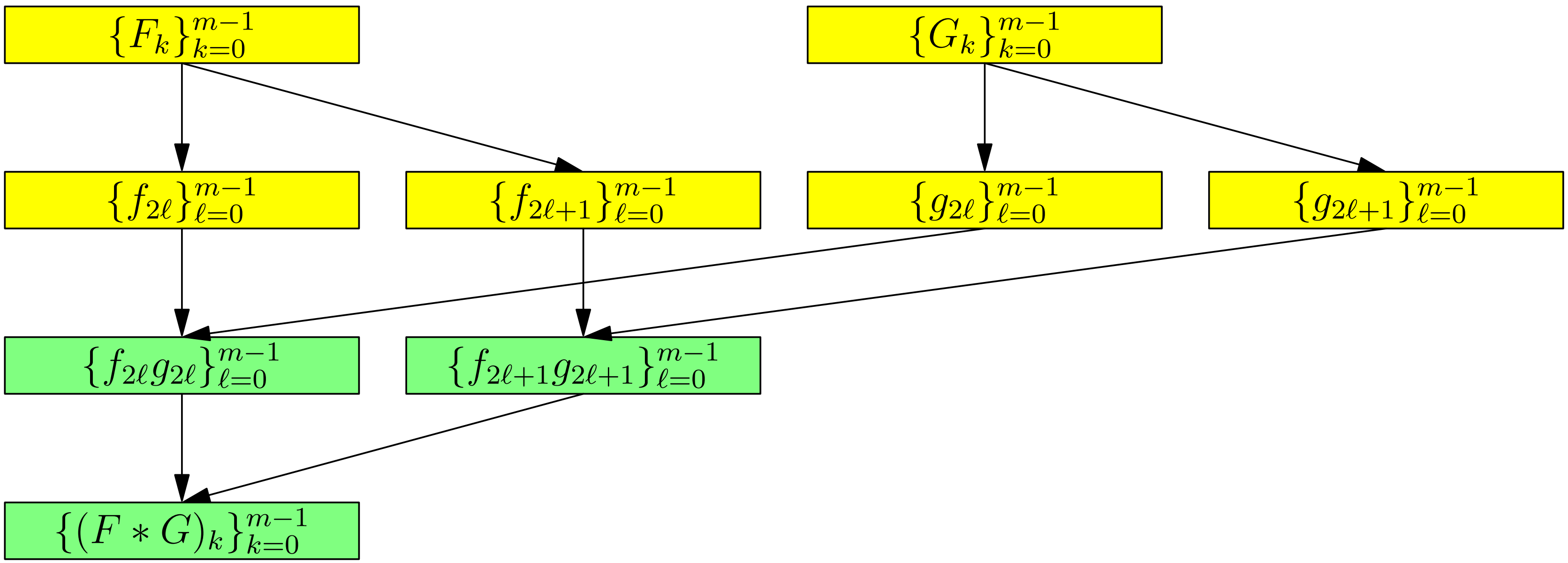
- Parallelized multidimensional implicit dealiasing routines have been implemented as a software layer **FFTW++** (v 2.05) on top of the **FFTW** library under the Lesser GNU Public License:

  `http://fftwpp.sourceforge.net/`



- Parallelized multidimensional implicit dealiasing routines have been implemented as a software layer **FFTW++** (v 2.05) on top of the **FFTW** library under the Lesser GNU Public License:

  `http://fftwpp.sourceforge.net/`



Full Fractional-Phase Fourier Transform

- We will also need the full fractional-phase Fourier transform

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

- On defining two sequences $g_j \doteq \zeta^{\alpha j^2/2}$ and $h_k \doteq g_k F_k$, this can be computed just like the partial Fourier transform, except the condition $g_{j-k} = 0$ when $j - k < 0$ is now replaced by $g_{j-k} = g_{k-j}$.

- This symmetry can be enforced by precomputing the inverse FFTs:

$$\{G_j\}_{j=0}^{N-1} = \mathtt{fft}^{-1}(\{\zeta^{-\alpha k^2/2} + \zeta^{-\alpha(k-N)^2/2}\}_{k=0}^{N-1}),$$

$$\{V_j\}_{j=0}^{N-1} = \mathtt{fft}^{-1}(\{\zeta_{2N}^{k}[\zeta^{-\alpha k^2/2} - \zeta^{-\alpha(k-N)^2/2}]\}_{k=0}^{N-1}).$$

- Then f_j can be calculated as the product of g_j and the jth output of the following function applied to the sequence $\{h_k\}_{k=0}^{N-1}$.

```
Input: vector f
Output: vector f
for k = 0 to N − 1 do
    u[k] ← ζ_{2N}^{k} f[k];
end
f ← fft(fft^{−1}(f) ∗ G);
u ← fft(fft^{−1}(u) ∗ V);
for k = 0 to N − 1 do
    f[k] ← f[k] + ζ_{2N}^{−k} u[k];
end
return f/(2N);
```

- In the above pseudocode, an asterisk $(*)$ denotes an element-by-element (vector) multiply.

Partial FFT: Special Case $c(j) = (pj + s)/q$

- Here p, q, and s are integers, with $p \neq 0$ and

$$f_j \doteq \sum_{k=0}^{\lfloor (pj+s)/q \rfloor} \zeta^{\alpha jk} F_k, \qquad j = 0, \ldots, M-1.$$

- Let $pj + s = qn + r$, with $n = 0, \ldots, N-1$. Then

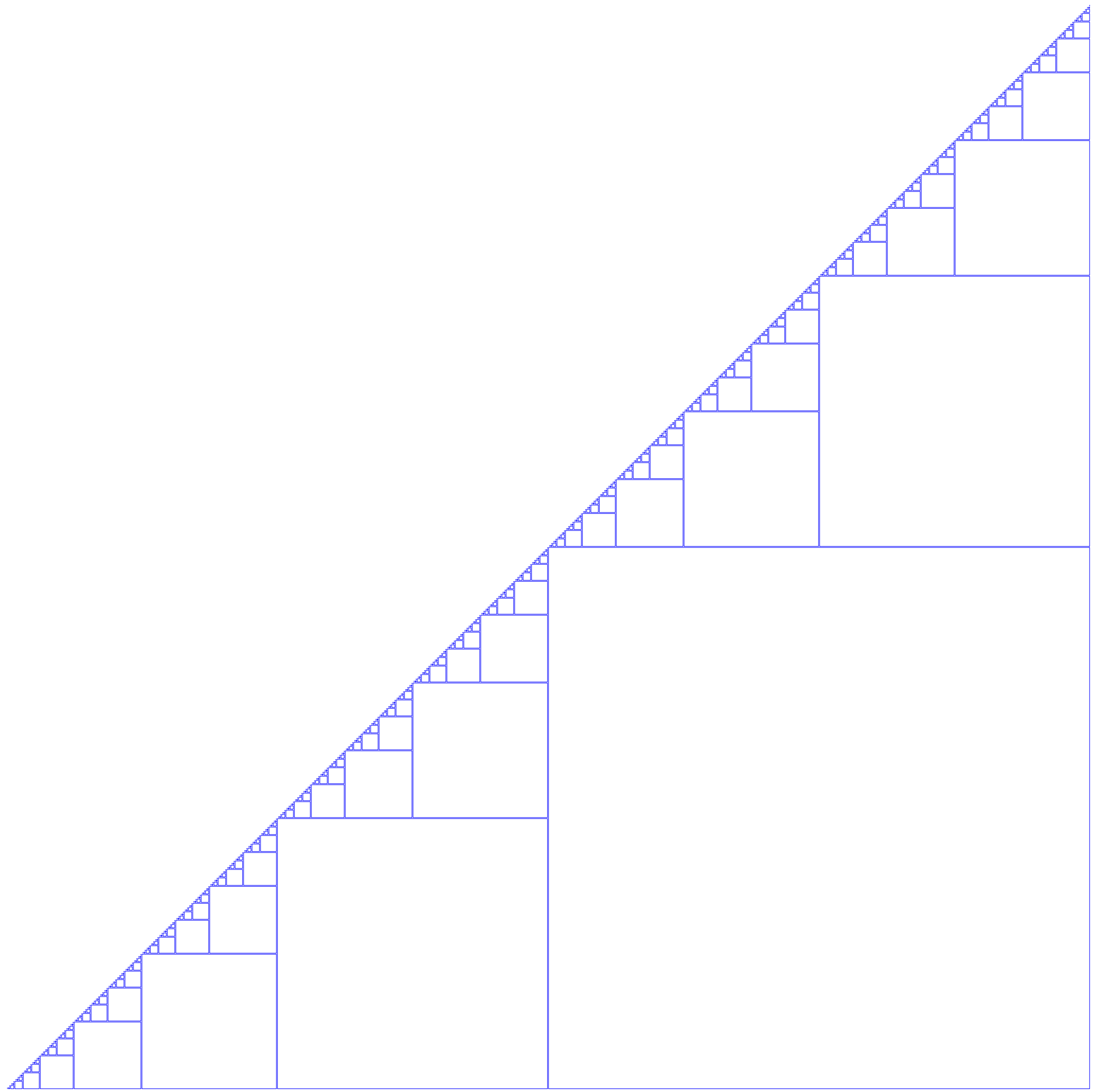
$$\begin{aligned} f_j &= \sum_{k=0}^{n} \zeta_p^{\alpha(qn+r-s)k} F_k \\ &= \sum_{k=0}^{n} \zeta_{2p}^{\alpha q\left[n^2+k^2-(n-k)^2\right]} \zeta_p^{\alpha(r-s)k} F_k \\ &= \zeta_{2p}^{\alpha qn^2} \sum_{k=0}^{n} \zeta_{2p}^{-\alpha q(n-k)^2} \zeta_{2p}^{\alpha qk^2} \zeta_p^{\alpha(r-s)k} F_k \end{aligned}$$

- On setting $g_k \doteq \zeta_{2p}^{\alpha q k^2}$ and $h_k \doteq g_k \zeta_p^{\alpha(r-s)k} F_k$, the result can be written as a convolution of two sequences $\{h_k\}$ and $\{g_k\}$:

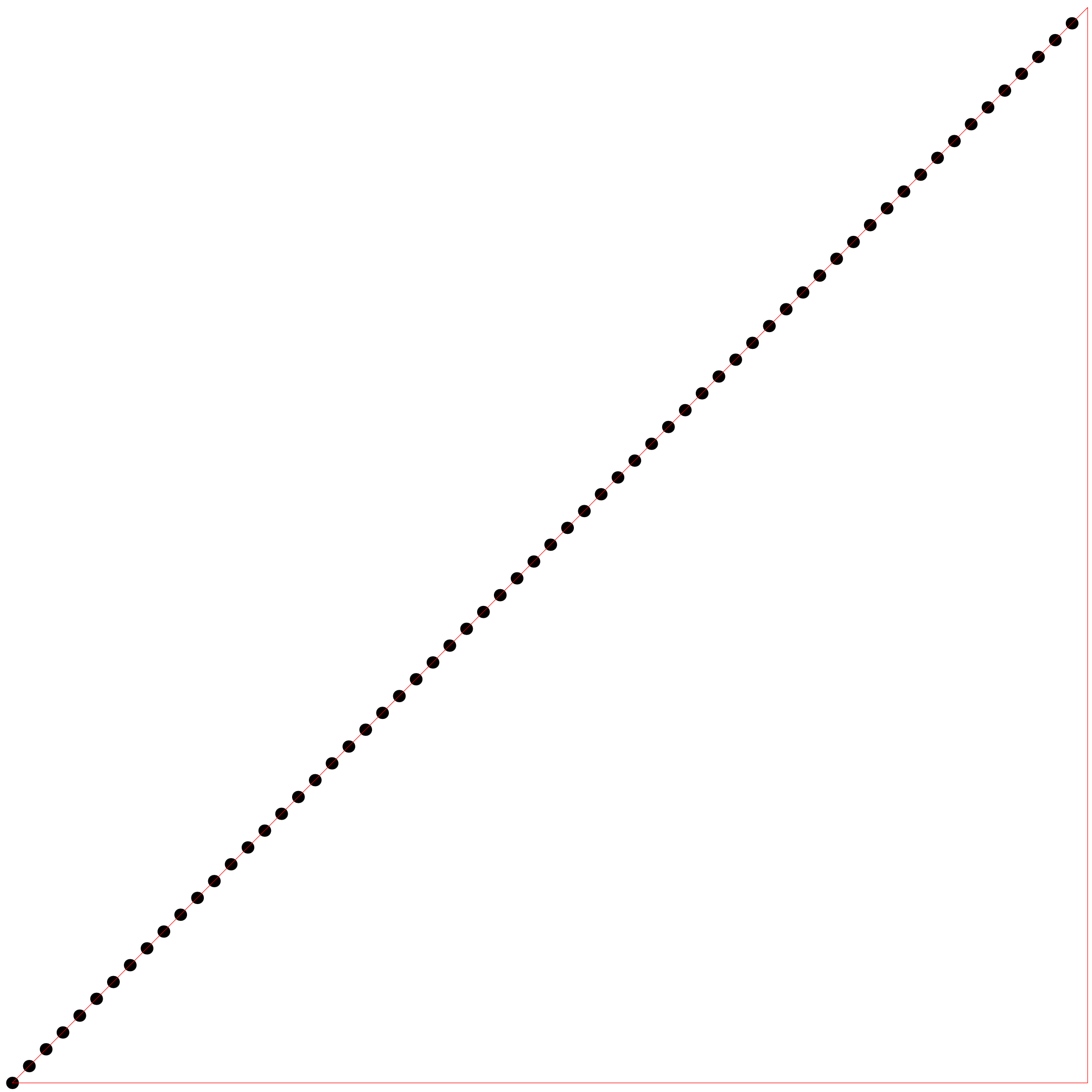
$$f_j = g_n \sum_{k=0}^{n} h_k \overline{g}_{n-k}, \qquad j = 0, \ldots, M-1.$$

- This general algorithm is only efficient when $p = 1$ or $q = 1$.
- The technique can be readily extended to higher dimensions.

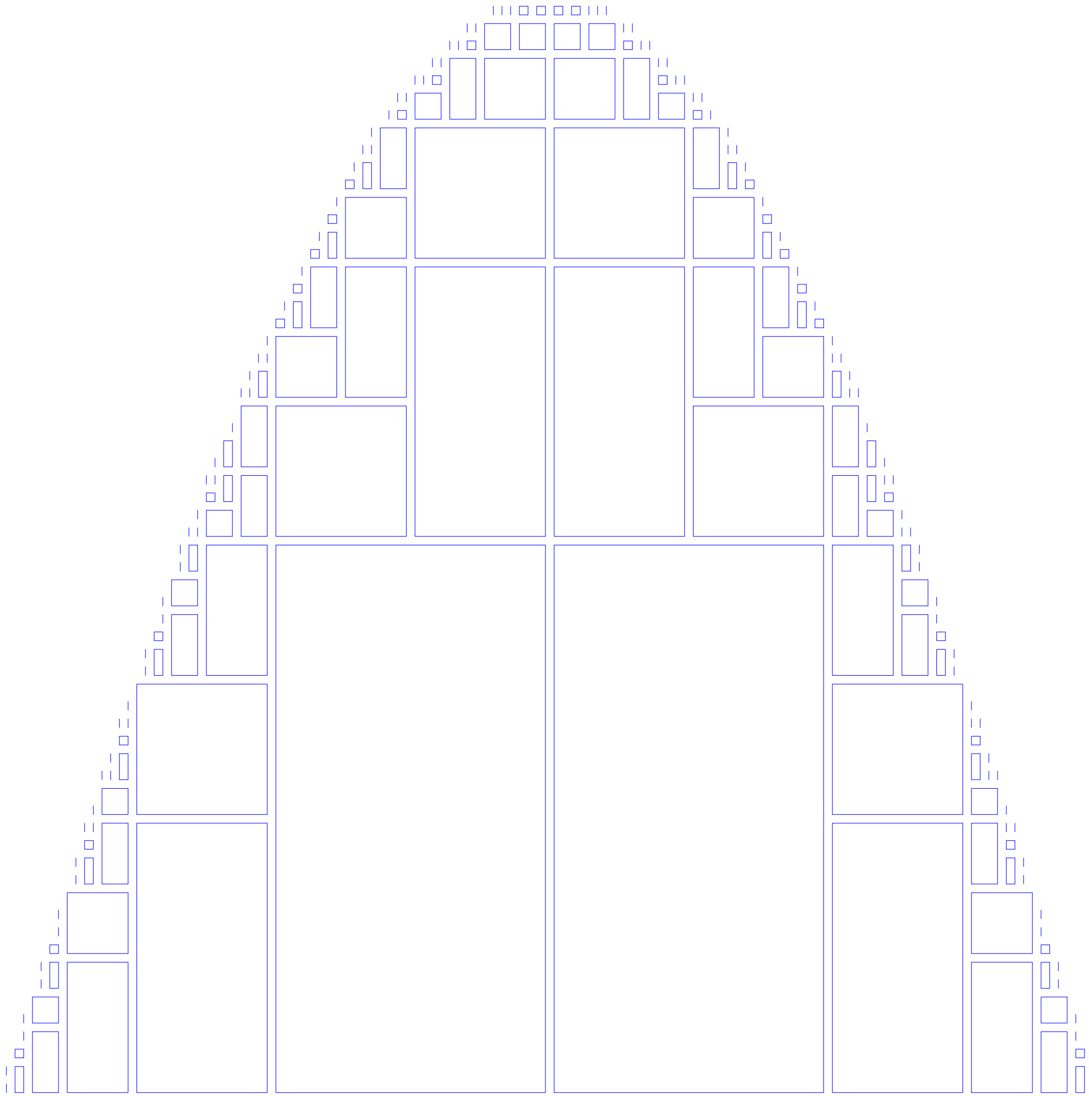
Rectangular subdivision for $c(j) = j$



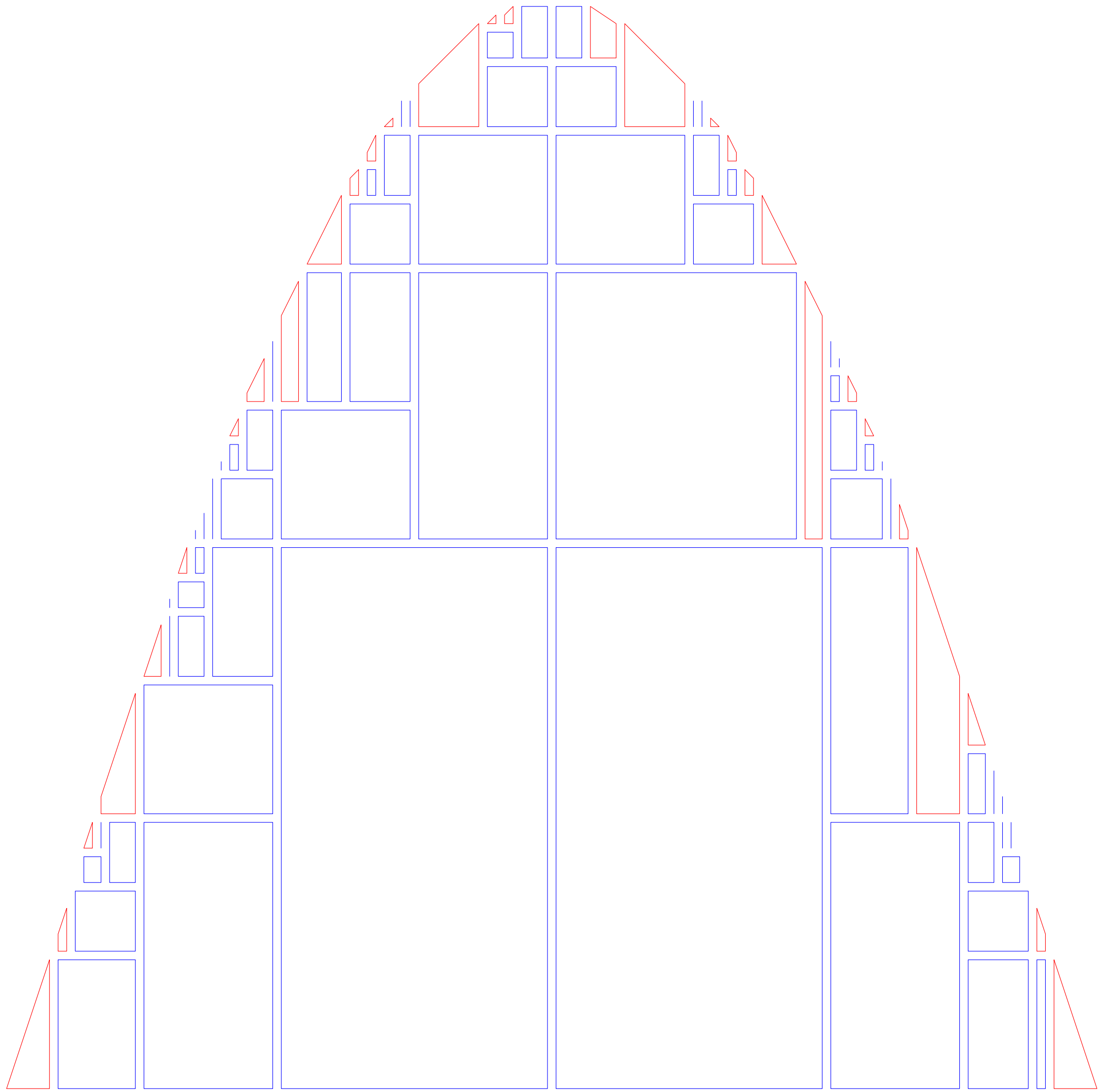
Triangular subdivision for $c(j) = j$



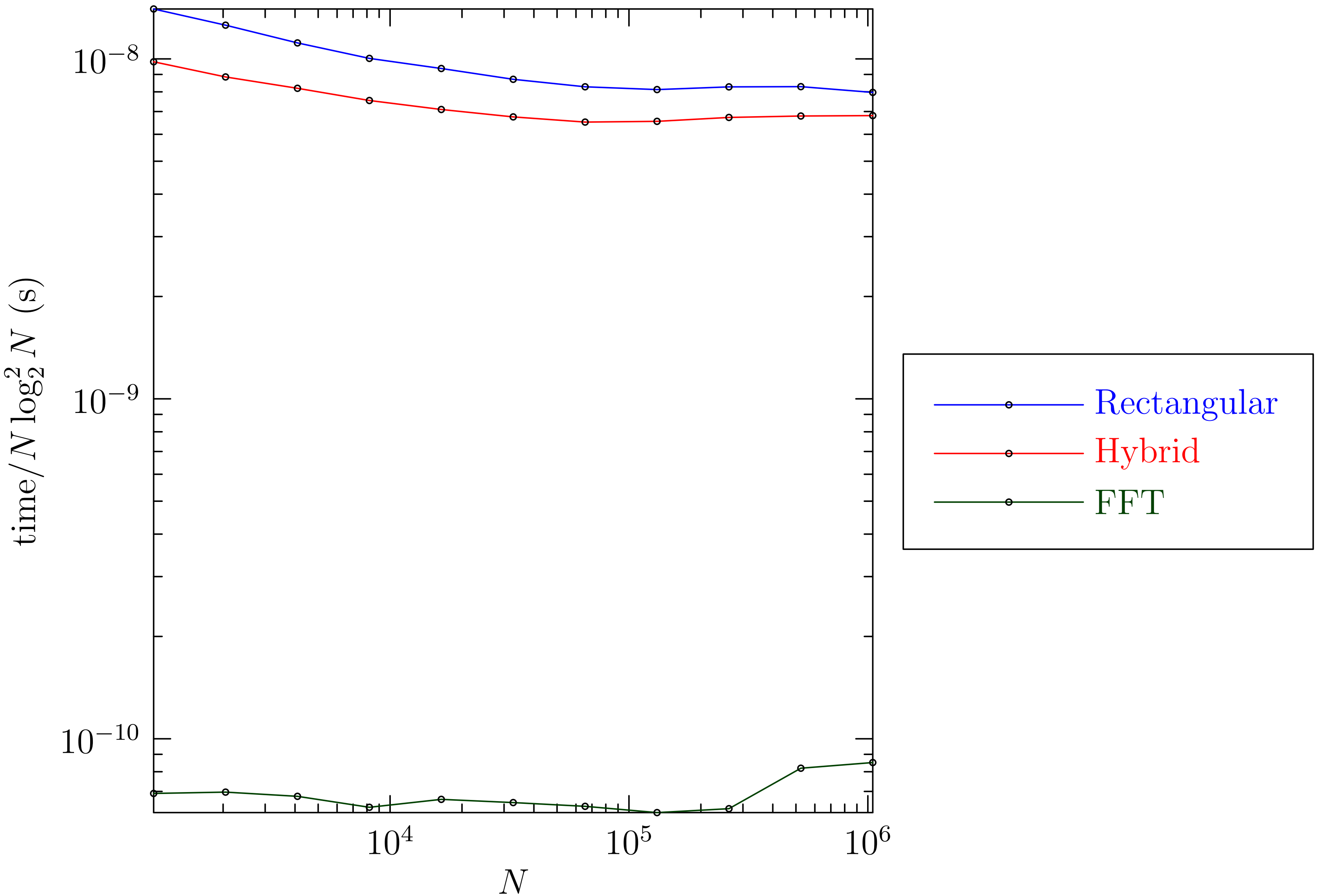
Rectangular: $c(j) = (N-1)\sin\frac{\pi j}{N-1}$



Hybrid: $c(j) = (N-1)\sin\frac{\pi j}{N-1}$



Computation time



Application: Kolmogorov Theory of Turbulence

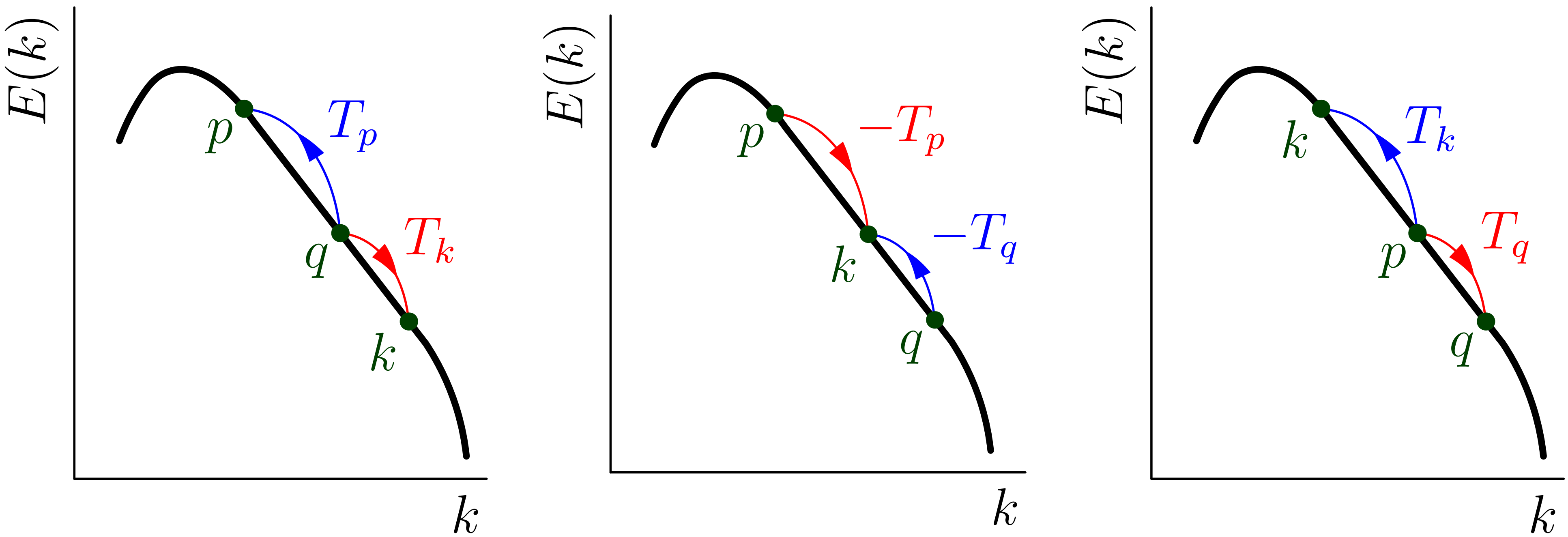
- Although the independence of the local inertial-range energy flux with wavenumber is one of the key hypothesis underlying Kolmogorov's famous 5/3 power-law form for the kinetic energy spectrum, it has never been directly tested, either experimentally or numerically.

- To validate Kolmogorov's uniform flux hypothesis in a high-resolution pseudospectral code, detailed wavenumber constraints must be imposed on the convolution.

- The key tool needed is the partial fast Fourier transform, where the summation limits are restricted by a spatially-dependent constraint.

- To this end, we have improved on previous attempts [Ying 2009] to develop a partial FFT based on the fractional-phase Fourier transform and Bluestein's algorithm [Bluestein 1970].

Flux Decomposition for a Single $(\boldsymbol{k}, \boldsymbol{p}, \boldsymbol{q})$ Triad



$L_k = T_k$
$S_k = 0$

$L_k = -T_p$
$S_k = -T_q$

$L_k = 0$
$S_k = T_k$

- Note that energy is conserved: $L_k + S_k = T_k = -T_p - T_q$. Thus

$$L_k = \mathrm{Re} \sum_{\substack{|\boldsymbol{k}|=k \\ |\boldsymbol{p}|<k \\ |\boldsymbol{k}-\boldsymbol{p}|<k}} M_{\boldsymbol{k},\boldsymbol{p}}\, \omega_{\boldsymbol{p}}\, \omega_{\boldsymbol{k}-\boldsymbol{p}}\, \omega_{\boldsymbol{k}}^* - \mathrm{Re} \sum_{\substack{|\boldsymbol{k}|=k \\ |\boldsymbol{p}|<k \\ |\boldsymbol{k}-\boldsymbol{p}|>k}} M_{\boldsymbol{p},\boldsymbol{k}-\boldsymbol{p}}\, \omega_{\boldsymbol{k}}\, \omega_{\boldsymbol{k}-\boldsymbol{p}}\, \omega_{\boldsymbol{p}}^*.$$

Conclusions

- A fast $\mathcal{O}(N \log N)$ algorithm for computing the partial fast Fourier transform is available, but with a relatively large coefficient.
- Improving on the work of Ying & Fomel [2009], we obtained a fast computational scaling, but with a smaller overall coefficient.
- The partial Fourier transform has applications in decomposing turbulent transfers into nonlocal and local fluxes.
- These techniques can be used to compute detailed inertial-range flux profiles and for the first time verify a key underpinning assumption of Kolmogorov's famous power-law conjecture for turbulence.
- This will allow us to verify and exploit inertial-range self-similarity in 2D turbulence and study the *flux locality profile*.

References

[Bluestein 1970] L. I. Bluestein, IEEE Trans. Audio and Electroacoustics, **18**:451, 1970.

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

[Roberts & Bowman 2017] M. Roberts & J. C. Bowman, Submitted to Journal of Computational Physics, 2017.

[Ying & Fomel 2009] L. Ying & S. Fomel, Multiscale Modeling and Simulation, **8**:110, 2009.