Butterfly Factorization

Yingzhou (Ryan) Li, Haizhao Yang, Eileen R. Martin,
Kenneth L. Ho, Lexing Ying

ICME & Math, Stanford

January 20, 2015

Outline

Butterfly Matrix

A matrix K of size $N \times N$ is a ***Butterfly Matrix*** if any contiguous $p \times q$ submatrix, $pq \leq N$, has a rank bounded by a constant independent of N.

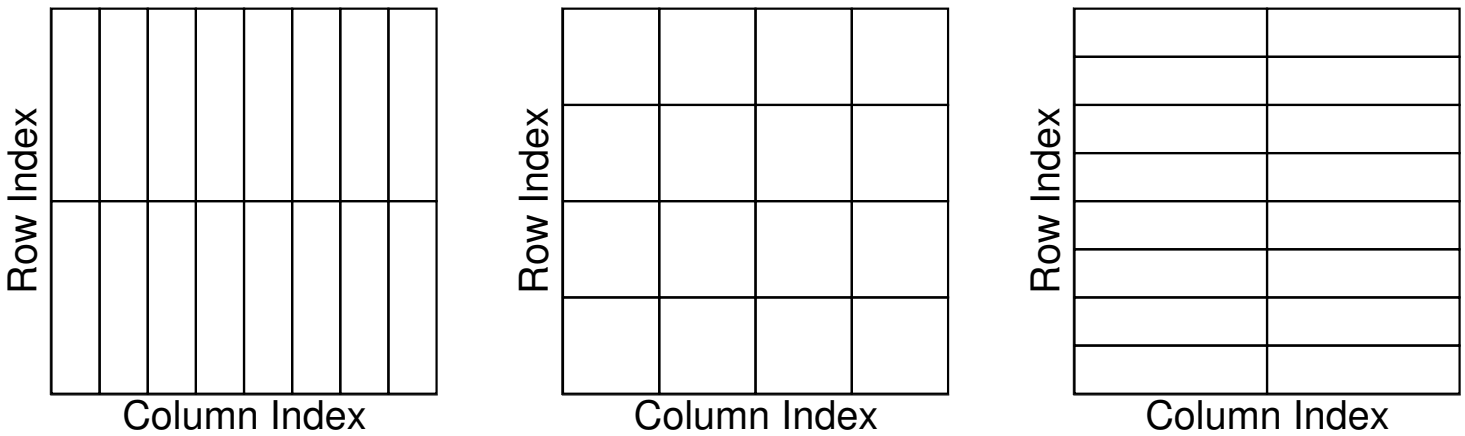

Figure : A butterfly matrix.

Fourier Integral Operators

$$(\mathcal{L}f)(x) = \int_{\mathbb{R}} e^{2\pi\imath\Phi(x,\xi)}\widehat{f}(\xi)d\xi,$$

where

- $\Phi(x,\xi)$ is a phase function and obeys a homogeneity condition of degree 1 in ξ, namely, $\Phi(x,\lambda\xi) = \lambda\Phi(x,\xi)$ for each $\lambda > 0$;
- $\widehat{f}$ is the Fourier transform of f.

The discrete FIO is defined as follows:

$$(Lf)(x) = \sum_{\xi\in\Omega} e^{2\pi\imath\Phi(x,\xi)}\widehat{f}(\xi)$$

for every $x \in X$, where $X = \left\{x = \frac{i-1}{N}, i = 1,2,\ldots,N\right\}$ and $\Omega = \left\{\xi = j - 1 - \frac{N}{2}, j = 1,2,\ldots,N\right\}$.

Composition of FIOs

Fact

The composition of FIOs is another FIO.

The discrete FIO is defined as follows:

$$(Lf)(x) = K\widehat{f} = \sum_{\xi \in \Omega} e^{2\pi\imath\Phi(x,\xi)}\widehat{f}(\xi).$$

Based on the property of FIO,

$$K_{new} = KFK$$

is a *Butterfly Matrix*, where F is the discrete Fourier transform matrix.

Special Functions

We denote the Hankel function of the first kind of order m by $H_m^{(1)}$. The Hankel function transform is defined as

$$u(x_i) = \sum_{j=1}^{N} H_{j-1}^{(1)}(x_i) g_j, \quad i = 1, 2, \ldots, N, \tag{1}$$

where the points x_i are defined via the formula,

$$x_i = N + \frac{2\pi}{3}(i-1). \tag{2}$$

The matrix

$$K = (k_{ij})_{i,j} = (H_{j-1}^{(1)}(x_i))_{i,j} \tag{3}$$

is a *Butterfly Matrix*.

Related Works

- $\mathcal{O}(N^2)$ algorithm
  - Dense matrix-vector multiplication
  - Precomputation of butterfly algorithm [O'Neil, 2007]
- $\mathcal{O}(N^{1.5} \log N)$ algorithm
  - Precomputation of Butterfly Factorization [Li et al., 2015]
- $\mathcal{O}(N^{1.25} \log N)$ algorithm
  - Wedge separation algorithm (2D) [Candès et al., 2007]
- $\mathcal{O}(N \log N)$ algorithm
  - Application of butterfly algorithm [O'Neil, 2007]
  - Polar butterfly algorithm [Candès et al., 2009, Poulson et al., 2014]
  - Multiscale butterfly algorithm [Li et al., 2014]
  - Application of Butterfly Factorization [Li et al., 2015]
  - Fast precomputation of Butterfly Factorization

Butterfly factorization

The butterfly factorization of K is

$$K \approx U^L G^{L-1} \cdots G^h M^h H^h \cdots H^{L-1} V^L,$$

where all factors are sparse matrices with $\mathcal{O}(N)$ nonzero entries, $L = \mathcal{O}(\log N)$ is the level number of domain trees and $h = L/2$.

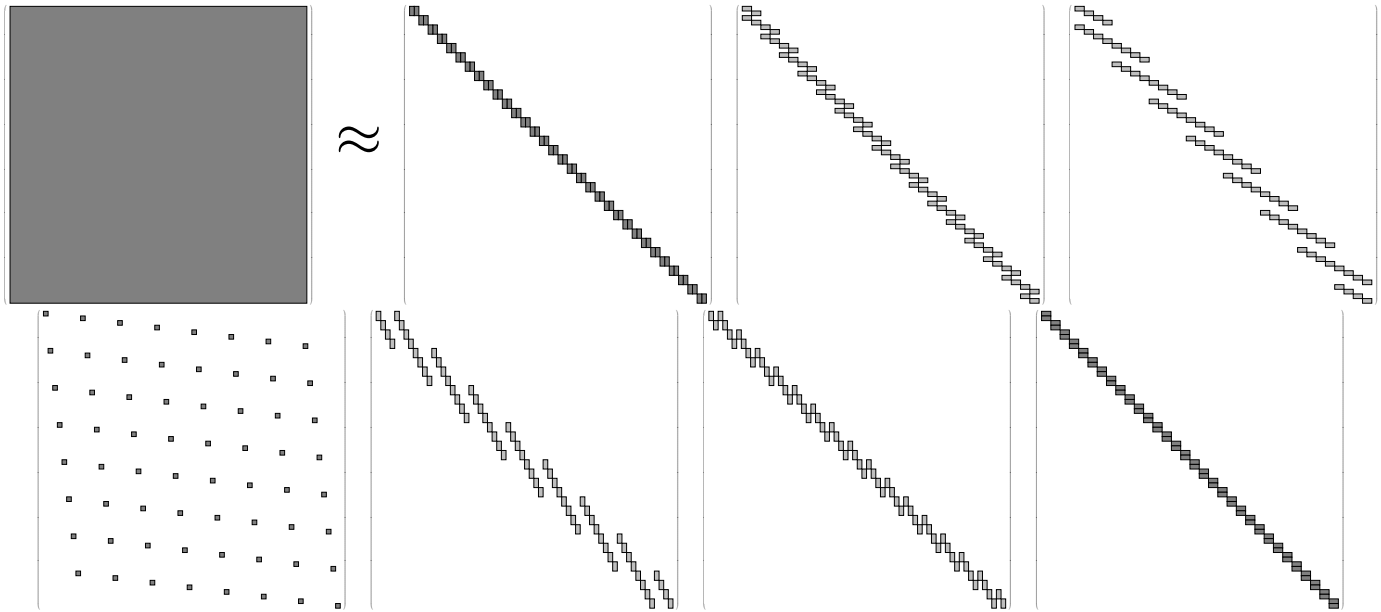

Figure : The butterfly factorization of a 64×64 butterfly matrix.

Middle level compression

By applying (quasi-) linear low-rank factorization method,

$$K \approx \begin{pmatrix} U^h_{1,1}V^h_{1,1} & U^h_{1,2}V^h_{1,2} & \cdots & U^h_{1,m}V^h_{1,m} \\ U^h_{2,1}V^h_{2,1} & U^h_{2,2}V^h_{2,2} & & U^h_{2,m}V^h_{2,m} \\ \vdots & & \ddots & \\ U^h_{m,1}V^h_{m,1} & U^h_{m,2}V^h_{m,2} & & U^h_{m,m}V^h_{m,m} \end{pmatrix}$$

$$= \begin{pmatrix} U^h_1 & & & \\ & U^h_2 & & \\ & & \ddots & \\ & & & U^h_m \end{pmatrix} M^h \begin{pmatrix} V^h_1 & & & \\ & V^h_2 & & \\ & & \ddots & \\ & & & V^h_m \end{pmatrix}$$

where $U^h_i = \begin{pmatrix} U^h_{i,1} & \cdots & U^h_{i,m} \end{pmatrix}$, $V^h_i = \begin{pmatrix} (V^h_{i,1})^* & \cdots & (V^h_{i,m})^* \end{pmatrix}^*$ and M^h is a permutation matrix.

Middle level compression

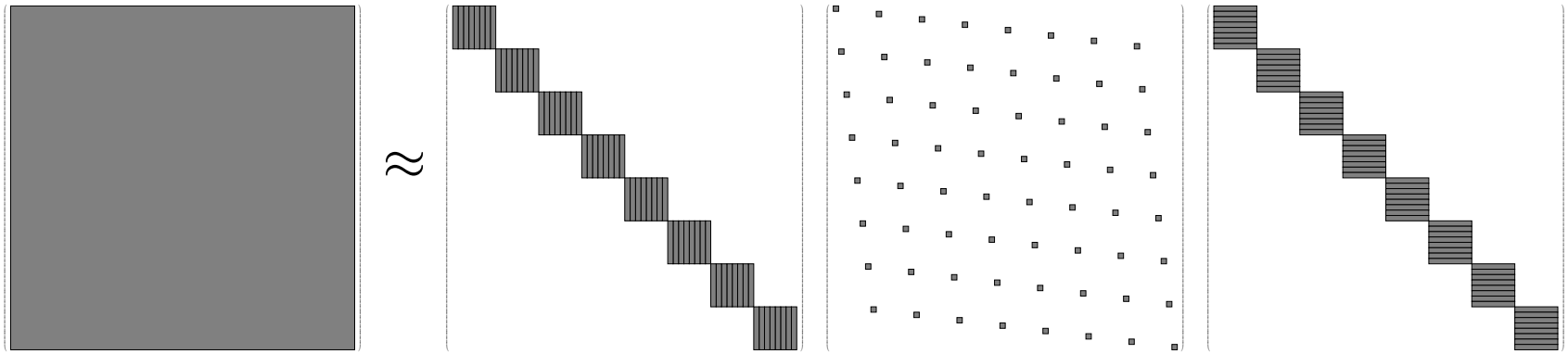

Figure : The middle level factorization of a 64×64 butterfly matrix $K \approx U^3 M^3 V^3$ assuming $r_\epsilon = 1$.

Recursive factorization of U^h

For $\ell = h, \ldots, L-1$, we split $U^\ell_{i,j}$ into top and bottom two parts, and factorize blocks of U^ℓ as follows,

$$\begin{pmatrix} U^\ell_{i,2j-1} & U^\ell_{i,2j} \end{pmatrix} = \begin{pmatrix} U^{\ell,t}_{i,2j-1} & U^{\ell,t}_{i,2j} \\ U^{\ell,b}_{i,2j-1} & U^{\ell,b}_{i,2j} \end{pmatrix} \approx \begin{pmatrix} U^{\ell+1}_{2i-1,j} G^\ell_{2i-1,j} \\ U^{\ell+1}_{2i,j} G^\ell_{2i,j} \end{pmatrix}.$$

Rearranging these factorization,

$$U^\ell \approx U^{\ell+1} G^\ell = \begin{pmatrix} U^{\ell+1}_1 & & & \\ & U^{\ell+1}_2 & & \\ & & \ddots & \\ & & & U^{\ell+1}_{2^{\ell+1}} \end{pmatrix} \begin{pmatrix} G^\ell_1 & & & \\ & G^\ell_2 & & \\ & & \ddots & \\ & & & G^\ell_{2^\ell} \end{pmatrix},$$

where $U^{\ell+1}_i = \begin{pmatrix} U^{\ell+1}_{i,1} & U^{\ell+1}_{i,2} & \cdots & U^{\ell+1}_{i,2^{L-\ell-1}} \end{pmatrix}$.

Recursive factorization of U^h

$$G_i^\ell = \begin{pmatrix} G^\ell_{2i-1,1} & & \\ & \ddots & \\ & & G^\ell_{2i-1,2^{L-\ell-1}} \\ \hline G^\ell_{2i,1} & & \\ & \ddots & \\ & & G^\ell_{2i,2^{L-\ell-1}} \end{pmatrix}$$

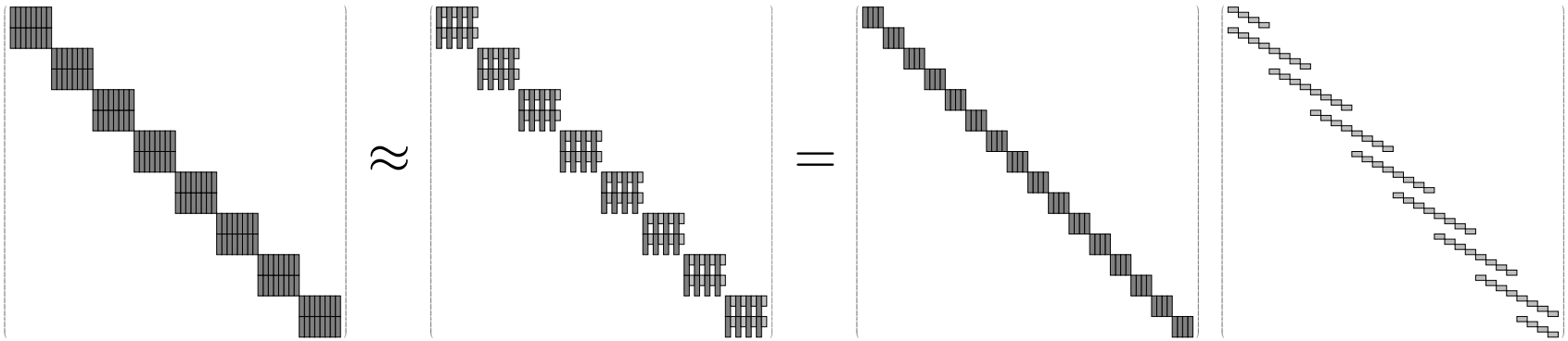

Figure : The recursive factorization of $U^3 = U^4 G^3$.

Recursive factorization

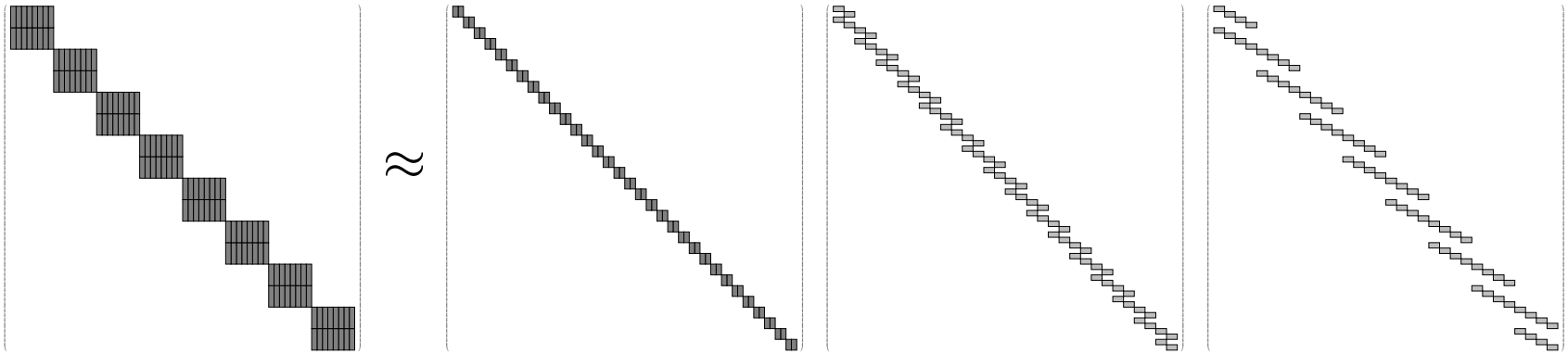

Figure : The recursive factorization of $U^3 \approx U^5 G^4 G^3$.

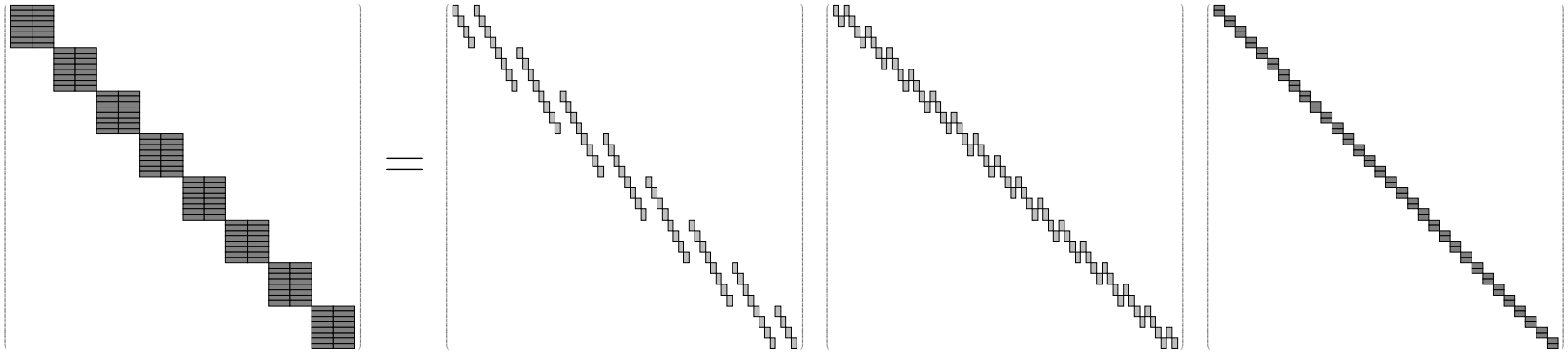

Figure : The recursive factorization $V^3 \approx H^3 H^4 V^5$.

Complexity

The complexity for the butterfly factorization is summarized as

	Randomized SVD	Randomized sampling
Middle level factorization	$\mathcal{O}(r_\epsilon C_K N^{1/2} + r_\epsilon^2 N^{3/2})$	$\mathcal{O}(r_\epsilon^2 N^{3/2})$
Recursive factorization	$\mathcal{O}(r_\epsilon^2 N^{3/2})$	
Total	$\mathcal{O}(r_\epsilon C_K N^{1/2} + r_\epsilon^2 N^{3/2})$	$\mathcal{O}(r_\epsilon^2 N^{3/2})$
Memory Complexity	$\mathcal{O}(r_\epsilon N^{3/2})$	$\mathcal{O}(r_\epsilon^2 N \log N)$
Application Complexity	$\mathcal{O}(r_\epsilon^2 N \log N)$	

Wave4@Math and Matlab

- Intel® Xeon® E7-4820 CPU @2.00GHz
- 8 Cores and 16 Threads
- 18M Cache
- 256 GB of memory
- OS: CentOS release 6.5 (Final)

- MATLAB® R2013a

Numerical results: FIO

The kernel is given by

$$\Phi(x,\xi) = x\cdot\xi + c(x)|\xi|,$$
$$c(x) = (2+\sin(2\pi x))/8.$$

N, r_ϵ	ϵ^a	$T_{Factor}(min)$	$T_a(sec)$	T_d/T_a	T_a/T_{FFT}
16384,4	1.37e-04	8.49e+00	1.60e-01	1.17e+02	3.49e+02
65536,4	1.37e-04	4.73e+01	7.38e-01	2.53e+02	2.40e+02
262144,4	1.71e-04	2.65e+02	3.21e+00	1.00e+03	2.39e+02
16384,6	4.85e-07	1.04e+01	2.58e-01	6.45e+01	5.64e+02
65536,6	6.80e-07	5.58e+01	1.27e+00	1.21e+02	4.15e+02
262144,6	1.38e-06	3.24e+02	6.24e+00	5.17e+02	4.64e+02
16384,8	1.34e-09	9.37e+00	4.57e-01	3.98e+01	9.99e+02
65536,8	3.96e-09	5.78e+01	2.43e+00	6.36e+01	7.91e+02
262144,8	2.02e-08	3.47e+02	1.10e+01	2.87e+02	8.21e+02

Numerical results: Special function

The Hankel function transform,

$$u(x_i) = \sum_{j=1}^{N} H_{j-1}^{(1)}(x_i) g_j, \quad i = 1, 2, \ldots, N. \tag{4}$$

N, r_ϵ	ϵ^a	$T_{Factor}(min)$	$T_a(sec)$	T_d/T_a	T_a/T_{FFT}
1024,4	1.28e-06	5.86e-01	6.83e-03	1.30e+02	4.00e+02
4096,4	8.27e-06	2.94e+00	2.68e-02	2.07e+02	3.01e+02
16384,4	6.73e-05	1.67e+01	1.04e-01	6.39e+02	2.28e+02
65536,4	2.92e-05	6.68e+01	5.37e-01	2.00e+03	1.82e+02
1024,6	7.55e-10	3.73e-01	8.94e-03	1.01e+02	5.23e+02
4096,6	2.27e-08	3.33e+00	4.07e-02	1.36e+02	4.57e+02
16384,6	2.72e-07	1.45e+01	3.09e-01	2.23e+02	6.80e+02
65536,6	8.12e-09	8.06e+01	1.56e+00	6.97e+02	5.29e+02

Numerical results: Composition of FIOs

The composition of FIOs is given as

$$K_{new} = KFK.$$

N, r_ϵ	ϵ^a	$T_{Factor}(min)$	$T_a(sec)$	T_d/T_a	T_a/T_{FFT}
4096,4	1.96e-02	4.20e+00	2.52e-02	2.62e+02	2.79e+02
16384,4	2.34e-02	4.65e+01	1.15e-01	3.25e+02	2.52e+02
65536,4	2.18e-02	4.33e+02	6.79e-01	5.49e+02	2.21e+02
4096,8	8.67e-05	4.94e+00	5.99e-02	1.10e+02	6.65e+02
16384,8	1.43e-04	6.23e+01	3.47e-01	1.08e+02	7.60e+02
65536,8	1.51e-04	6.91e+02	1.76e+00	2.12e+02	5.74e+02
4096,12	1.05e-07	6.35e+00	1.12e-01	5.88e+01	1.25e+03
16384,12	2.55e-07	7.58e+01	7.64e-01	4.91e+01	1.67e+03
65536,12	2.69e-07	7.63e+02	4.39e+00	8.49e+01	1.43e+03

Conclusion

- The butterfly factorization can be applied to all kernels with butterfly property.
- The factorization reduces the complexity of original butterfly algorithm whose precomputation complexity is $\mathcal{O}(N^2)$.
- The factorization complexity is $\mathcal{O}(N^{1.5} \log N)$.
- The application complexity is $\mathcal{O}(N \log N)$ with a tiny pre-factor.
- This butterfly factorization can be easily extended to higher dimensions..

Q & A

Suggested Questions:

- What is the memory complexity and practical usage?
- How to extend the algorithm to higher dimensions?

The End

Thank You!

Selected References I


Candès, E., Demanet, L., and Ying, L. (2007).
Fast computation of Fourier integral operators.
SIAM Journal on Scientific Computing, 29(6):2464–2493.


Candès, E., Demanet, L., and Ying, L. (2009).
A fast butterfly algorithm for the computation of Fourier integral operators.
Multiscale Modeling and Simulation, 7(4):1727–1750.


Li, Y., Yang, H., Ho, K. L., and Ying, L. (2015).
Butterfly factorization.


Li, Y., Yang, H., and Ying, L. (2014).
Multiscale butterfly algorithm for fourier integral operators.


O'Neil, M. P. (2007).
A New Class of Analysis-based Fast Transforms.
PhD thesis, New Haven, CT, USA.
AAI3293360.

Selected References II

Poulson, J., Demanet, L., Maxwell, N., and Ying, L. (2014).

A parallel butterfly algorithm.

SIAM Journal on Scientific Computing, 36(1):C49–C65.