The Diagonal Algorithm for multidimensional discrete Fourier transforms

Þorgeir Sigurðsson
Stefán Ingi Valdimarsson
Sven Þ. Sigurðsson

University of Edinburgh

4 September 2013

A story

- Joint work with
  - Þorgeir Sigurðsson at the Icelandic Radiation Safety Authority
  - Sven Þ. Sigurðsson at the University of Iceland (Emeritus)
- The diagonal algorithm for *m*-dimensional discrete Fourier transforms (DFT)

A story

- Joint work with
  - Þorgeir Sigurðsson at the Icelandic Radiation Safety Authority
  - Sven Þ. Sigurðsson at the University of Iceland (Emeritus)
- The diagonal algorithm for *m*-dimensional discrete Fourier transforms (DFT)
  - Simple — based on the Cooley–Tukey method
  - Fast — reduces number of multiplications by a factor of *m* compared with row–column method (asymptotically)
  - Interesting — analysis of operation count

A story

- Joint work with
  - Þorgeir Sigurðsson at the Icelandic Radiation Safety Authority
  - Sven Þ. Sigurðsson at the University of Iceland (Emeritus)
- The diagonal algorithm for *m*-dimensional discrete Fourier transforms (DFT)
  - Simple — based on the Cooley–Tukey method
  - Fast — reduces number of multiplications by a factor of *m* compared with row–column method (asymptotically)
  - Interesting — analysis of operation count
- Caveat — Polynomial methods are fast(er) (Nussbaumer–Quandalle), also Bernadini
  - Few implementations exist

The Cooley–Tukey method (one dimension)

- DFT of a vector (x_i) of length N a power of 2. $(\omega = e^{-j2\pi/N})$

$$\hat{x}_k = \sum_{i=0}^{N-1} x_i \omega_N^{ik}$$

The Cooley–Tukey method (one dimension)

- DFT of a vector (x_i) of length N a power of 2. $(\omega = e^{-j2\pi/N})$

$$\hat{x}_k = \sum_{i=0}^{N-1} x_i \omega_N^{ik} = \sum_{i=0}^{N/2-1} x_{2i}\omega_{N/2}^{ik} + \omega_N^k \sum_{i=0}^{N/2-1} x_{2i+1}\omega_{N/2}^{ik}$$

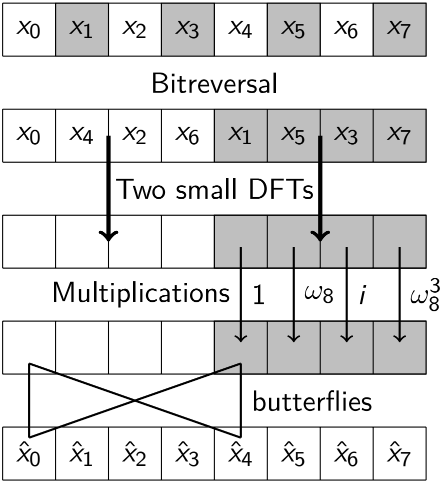
The Cooley–Tukey method (one dimension)

- DFT of a vector (x_i) of length N a power of 2. ($\omega = e^{-j2\pi/N}$)

$$\hat{x}_k = \sum_{i=0}^{N-1} x_i \omega_N^{ik} = \sum_{i=0}^{N/2-1} x_{2i}\omega_{N/2}^{ik} + \omega_N^k \sum_{i=0}^{N/2-1} x_{2i+1}\omega_{N/2}^{ik}$$

$$\hat{x}_{N/2+k} = \sum_{i=0}^{N/2-1} x_{2i}\omega_{N/2}^{ik} - \omega_N^k \sum_{i=0}^{N/2-1} x_{2i+1}\omega_{N/2}^{ik}$$

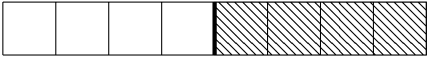
The Cooley–Tukey method graphically



$$N = 2^k, \qquad M_k = 2M_{k-1} + 2^{k-1}, \qquad M_k = k2^{k-1} = \frac{1}{2} N \lg(N)$$

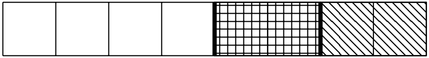
Radix 2 and split radix

- Real multiplications: one complex $=$ three real
- Radix 2



Multiplications $\frac{3}{2}N\lg(N) - 5N + 8$

- Split radix



Multiplications $N\lg(N) - 3N + 4$

The Row–Column and the Vector algorithms

- The Row–Column method

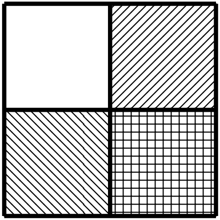
$$\hat{x}_{k_1,k_2} = \sum_{i_1=0}^{N-1} \sum_{i_2=0}^{N-1} x_{i_1,i_2} \omega_N^{i_1 k_1 + i_2 k_2} = \sum_{i_2=0}^{N-1} \left(\sum_{i_1=0}^{N-1} x_{i_1,i_2} \omega_N^{i_1 k_1} \right) \omega_N^{i_2 k_2}$$

The Row–Column and the Vector algorithms

- The Row–Column method

$$\hat{x}_{k_1,k_2} = \sum_{i_1=0}^{N-1}\sum_{i_2=0}^{N-1} x_{i_1,i_2}\omega_N^{i_1k_1+i_2k_2} = \sum_{i_2=0}^{N-1}\left(\sum_{i_1=0}^{N-1} x_{i_1,i_2}\omega_N^{i_1k_1}\right)\omega_N^{i_2k_2}$$

- The Vector method



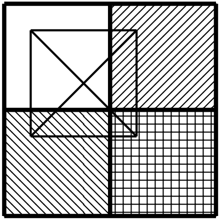
  - DFT in the small squares, multiplication in shaded ones

The Row–Column and the Vector algorithms

- The Row–Column method

$$\hat{x}_{k_1,k_2} = \sum_{i_1=0}^{N-1} \sum_{i_2=0}^{N-1} x_{i_1,i_2} \omega_N^{i_1 k_1 + i_2 k_2} = \sum_{i_2=0}^{N-1} \left(\sum_{i_1=0}^{N-1} x_{i_1,i_2} \omega_N^{i_1 k_1} \right) \omega_N^{i_2 k_2}$$

- The Vector method



  - DFT in the small squares, multiplication in shaded ones
  - Then use butterflies

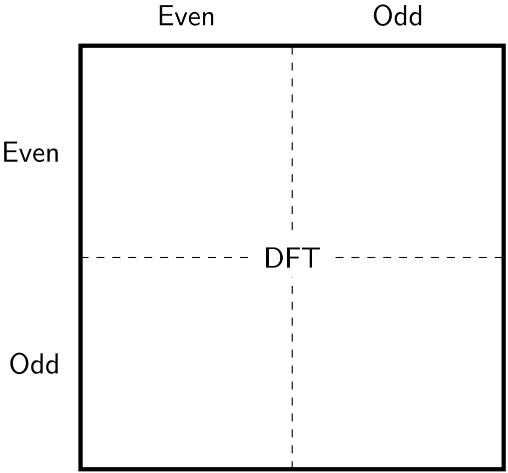
Real multiplication count

	Radix 2	Split radix
1-D	$\frac{3}{2} N \lg(N) + O(N)$	$N \lg(N) + O(N)$
2-D row–col	$\frac{3}{2} N \lg(N) + O(N)$	$N \lg(N) + O(N)$
2-D vector	$\frac{9}{8} N \lg(N) + O(N)$	$\frac{9}{14} N \lg(N) + O(N)$
2-D diagonal	$\frac{3}{4} N \lg(N) + O(N\sqrt{\lg(N)})$	$\frac{1}{2} N \lg(N) + O(N\sqrt{\lg(N)})$

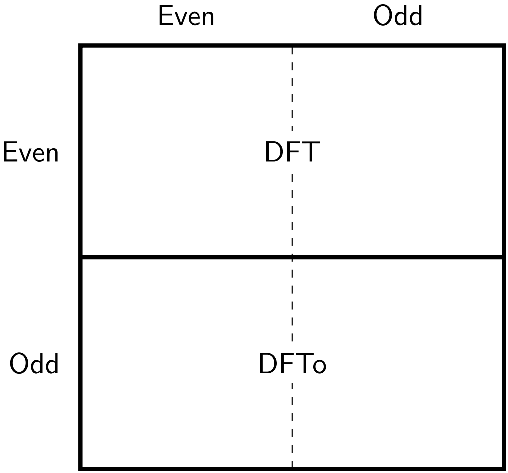
Total operation count: $A + 2M = 2N \lg(N) + 2M$.

The Diagonal Algorithm

- Pass more efficiently along the dimensions.

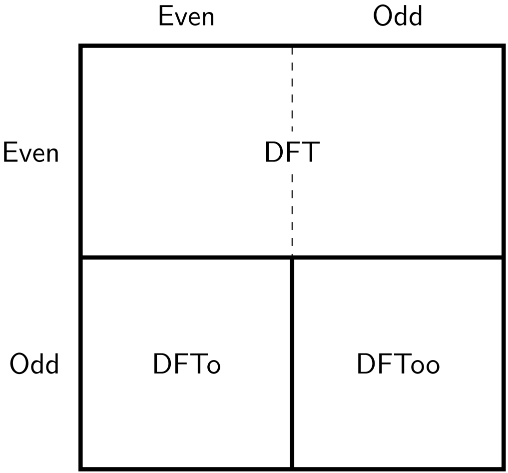


- Pass more efficiently along the dimensions.



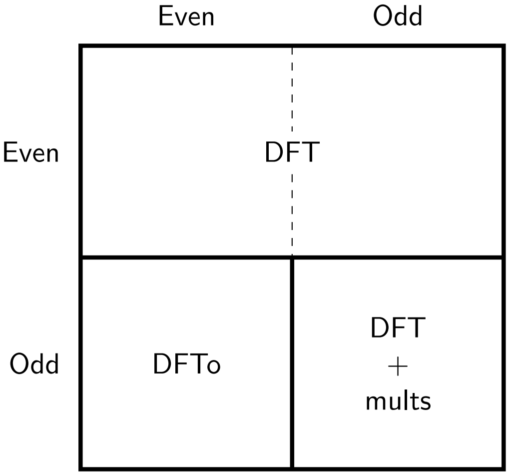
The Diagonal Algorithm

- Pass more efficiently along the dimensions.



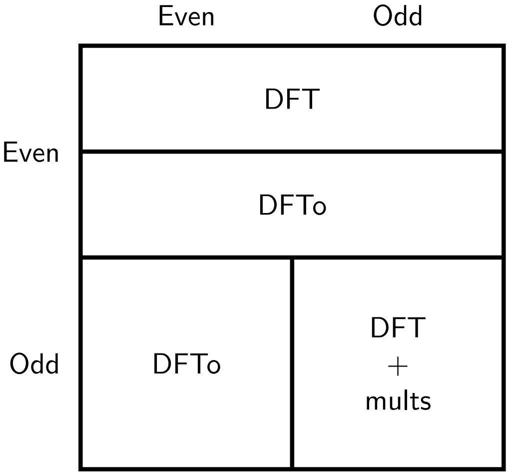
The Diagonal Algorithm

- Pass more efficiently along the dimensions.



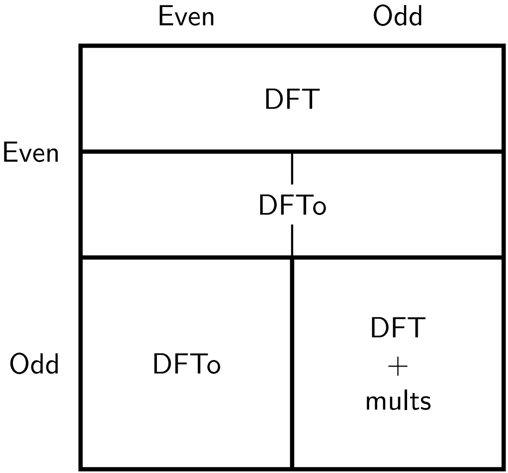
The Diagonal Algorithm

- Pass more efficiently along the dimensions.



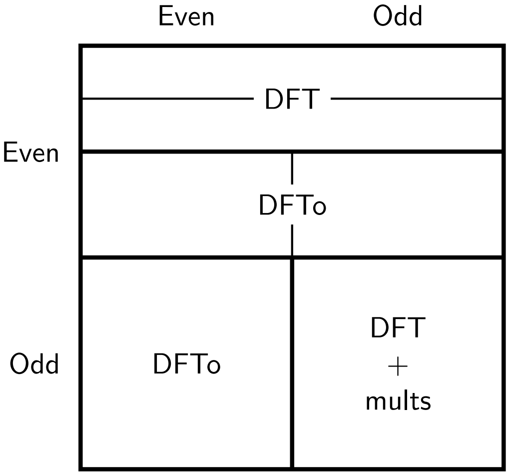
The Diagonal Algorithm

- Pass more efficiently along the dimensions.

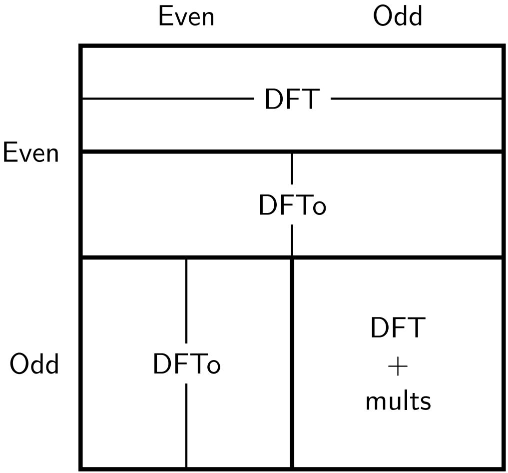


The Diagonal Algorithm

- Pass more efficiently along the dimensions.

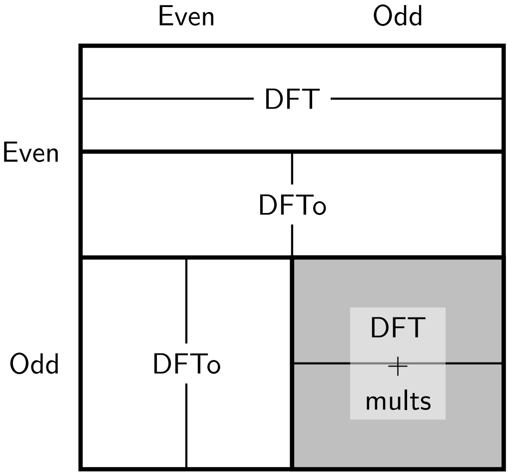


- Pass more efficiently along the dimensions.



The Diagonal Algorithm

- Pass more efficiently along the dimensions.



Multiplication count

- Complex multiplications for matrix of size $N = N_1 \times N_2 = 2^{k_1} \times 2^{k_2}$.

$$\begin{aligned} M(k_1, k_2) &= M(k_1 - 1, k_2) + Mo(k_1 - 1, k_2), \\ Mo(k_1 - 1, k_2) &= Mo(k_1 - 1, k_2 - 1) + Moo(k_1 - 1, k_2 - 1), \\ Moo(k_1 - 1, k_2 - 1) &= M(k_1 - 1, k_2 - 1) + 2^{k_1 - 1} 2^{k_2 - 1} \end{aligned}$$

- Reduces to

$$M(k_1, k_2) = M(k_1 - 1, k_2) + M(k_1, k_2 - 1) + 2^{k_1 - 1} 2^{k_2 - 1}$$

- Boundary conditions

$$M(k, 0) = M(0, k) = M(k) = k 2^{k-1}$$

Analysis

$$M(k_1, k_2) = M(k_1 - 1, k_2) + M(k_1, k_2 - 1) + 2^{k_1 - 1} 2^{k_2 - 1}$$
$$M(k, 0) = M(0, k) = k 2^{k-1}$$

- $M_1(k_1, k_2) = (k_1 + k_2) 2^{k_1 + k_2 - 2}$ solves inhomogeneous part
- Then $M = M_1 + M_2$ with

$$M_2(k_1, k_2) = M_2(k_1 - 1, k_2) + M_2(k_1, k_2 - 1)$$
$$M_2(k, 0) = M_2(0, k) = k 2^{k-2}$$

- Generating function for M_2

$$\sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} M_2(k_1, k_2) x^{k_1} y^{k_2} = \frac{1}{2(1 - x - y)} \left(\frac{x(1-x)}{(1-2x)^2} + \frac{y(1-y)}{(1-2y)^2} \right)$$

Diagonal coefficients

- DFT of a square matrix $N = N_1 \times N_1 = 2^k \times 2^k$
- Then $M(k,k) = k2^{2k-1} + M_2(k,k) = \frac{1}{4} N \lg(N) + M_2(k,k)$
- Generating function for M_2 is

$$F_2(x,y) = \frac{1}{2(1-x-y)} \left(\frac{x(1-x)}{(1-2x)^2} + \frac{y(1-y)}{(1-2y)^2} \right)$$

- Diagonal method

$$G_2(x) = \frac{1}{2\pi i} \oint \frac{F_2(x/\tau, \tau)}{\tau} d\tau = \sum_{k=0}^{\infty} M_2(k,k) x^k$$

- $G_2(x) = \dfrac{x}{(1-4x)^{3/2}} = \displaystyle\sum_{k=0}^{\infty} \frac{k}{2} \binom{2k}{k} x^k$

Real multiplications

	Radix 2	Split radix
$G(x)$	$\frac{16x^3(1+2x)(2+\sqrt{1-4x})}{(1-4x)^2(1-2x)}$	$\frac{16x^3(\sqrt{1-4x}+2\sqrt{(1-x)(1+3x+4x^2)})}{(1-4x)^2(1-2x)\sqrt{(1-x)(1+3x+4x^2)}}$
$G(x)$	$\frac{3}{2}\frac{1}{(1-4x)^2}+\frac{3}{4}\frac{1}{(1-4x)^{3/2}}+\dots$	$\frac{1}{(1-4x)^2}+\frac{1}{\sqrt{6}}\frac{1}{(1-4x)^{3/2}}+\dots$
$M(N)$	$\frac{3}{4}N\lg(N)+\frac{3}{2\sqrt{2\pi}}N\sqrt{lg(N)}$	$\frac{1}{2}N\lg(N)+\frac{1}{\sqrt{3\pi}}N\sqrt{lg(N)}$

Numbers in two dimensions

- Real multiplications

k $(N \times N = 2^k \times 2^k)$	3	5	7	9	11
Radix 2 Row-Col	64	5,632	182,272	4,464,640	96,501,760
Radix 2 Vector	48	4,224	136,704	3,348,480	72,376,320
Radix 2 Diagonal	48	3,808	116,064	2,741,952	57,853,536
Split Radix Row-Col	64	4,352	132,096	3,149,824	67,125,248
Split Radix Vector	48	2,928	87,024	2,058,096	43,676,400
Split Radix Diagonal	48	2,800	80,624	1,865,200	38,963,440

Numbers in two dimensions

- Real multiplications

k $(N \times N = 2^k \times 2^k)$	3	5	7	9	11
Radix 2 Row-Col	64	5,632	182,272	4,464,640	96,501,760
Radix 2 Vector	48	4,224	136,704	3,348,480	72,376,320
Radix 2 Diagonal	48	3,808	116,064	2,741,952	57,853,536
Radix 2 NQ	48	3,600	106,512	2,490,384	52,428,816
Split Radix Row-Col	64	4,352	132,096	3,149,824	67,125,248
Split Radix Vector	48	2,928	87,024	2,058,096	43,676,400
Split Radix Diagonal	48	2,800	80,624	1,865,200	38,963,440
Split Radix NQ	48	2,736	76,464	1,747,632	36,350,640

- NQ is the algorithm of Nussbaumer–Quandalle

Higher dimensions

- Complex multiplications in three dimensions

$$\begin{aligned} M(k_1,k_2,k_3) =& M(k_1-1,k_2,k_3)+M(k_1,k_2-1,k_3) \\ &+M(k_1,k_2,k_3-1)-M(k_1-1,k_2-1,k_3) \\ &-M(k_1-1,k_2,k_3-1)-M(k_1,k_2-1,k_3-1) \\ &+2M(k_1-1,k_2-1,k_3-1)+2^{k_1+k_2+k_3-3} \\ M(k_1,k_2,0) =& M(k_1,0,k_2)=M(0,k_1,k_2)=M(k_1,k_2) \end{aligned}$$

- Particular solution to inhomogeneous problem

$$M_1(k_1,k_2,k_3)=(k_1+k_2+k_3)2^{k_1+k_2+k_3-1}/3$$

- Generating function or $M_2 = M - M_1$.

$$\frac{G(x,y,z)}{(1-2x)^2\cdots(1-x-y)\cdots(1-x-y-z+xy+xz+yz-2xyz)}$$

Addendum

- Asymptotics of multivariate sequences
- *Analytic Combinatorics in Several Variables* by Pemantle and Wilson (2013)

$$F(x,y) = \sum_{r=0}^{\infty}\sum_{s=0}^{\infty} a_{r,s} x^r y^s$$

$$a_{r,s} = \frac{1}{(2\pi i)^2} \oint \oint \frac{F(x,y)}{x^{r+1} y^{s+1}} dxdy$$

- Asymptotics in a fixed direction $\mathbf{r} = (r,s) = |\mathbf{r}|(\hat{r},\hat{s})$ as $|\mathbf{r}| \to \infty$

Addendum

- Our function

$$a_{r,s} = \frac{1}{(2\pi i)^2} \oint \oint \frac{G(x,y)}{(1-2x)^2(1-2y)^2(1-x-y)x^{r+1}y^{s+1}} dydx$$

Addendum

- Our function

$$a_{r,s} = \frac{1}{(2\pi i)^2} \oint \oint \frac{G(x,y)}{(1-2y)^2(1-x-y)^3 x^{r+1} y^{s+1}} dydx$$

- $|x^r y^s| = \exp(r \log|x| + s \log|y|)$
- For a fixed direction $(\hat{r}, \hat{s})$ look for points where $H(x,y) = (1-2y)^2(1-x-y)^3 = 0$ and $r \log|x| + s \log|y|$ is minimized.
- Smooth points vs. mutliple points