FFTs in Graphics and Vision

The Fast Fourier Transform

Outline

The FFT Algorithm

Multi-Dimensional FFTs

More Applications

Real FFTs

Computational Complexity

To compute the correlation of two periodic, n-dimensional arrays $\mathbf{f},\mathbf{g}\in\mathbb{C}^n$:

1. Express $\mathbf{f}$ and $\mathbf{g}$ in the basis $\{\mathbf{z}^0,\ldots,\mathbf{z}^{n-1}\}\subset\mathbb{C}^n$:

$$\mathbf{f}=\sum_{k=0}^{n-1}\hat{\mathbf{f}}_k\cdot\mathbf{z}^k\quad\text{and}\quad\mathbf{g}=\sum_{k=0}^{n-1}\hat{\mathbf{g}}_k\cdot\mathbf{z}^k$$

2. Multiply (and scale) the coefficients:

$$(\mathbf{f}\star\mathbf{g})=\sqrt{2\pi}\cdot\sum_{k=0}^{n-1}\hat{\mathbf{f}}_k\cdot\bar{\hat{\mathbf{g}}}_k\cdot\mathbf{z}^{-k}$$

3. Evaluate at every index j:

$$(\mathbf{f}\star\mathbf{g})_j=\sqrt{2\pi}\cdot\sum_{k=0}^{n-1}\hat{\mathbf{f}}_k\cdot\bar{\hat{\mathbf{g}}}_k\cdot\mathbf{z}_j^{-k}$$

Goal

Given $\mathbf{g} \in \mathbb{C}^n$, we would like to compute the Fourier coefficients $\hat{\mathbf{g}}$:

$$\mathbf{g} = \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot \mathbf{z}^k$$

with $\mathbf{z}^k$ the normalized discrete samples of the complex exponentials at n regular positions:

$$\mathbf{z}^k = \left(\frac{e^{ik\theta_0}}{\sqrt{2\pi}}, \cdots, \frac{e^{ik\theta_{n-1}}}{\sqrt{2\pi}} \right)$$

Challenge

How can we compute all n Fourier coefficients efficiently?

Challenge

How can we compute all n Fourier coefficients?

Challenge

How can we compute all n Fourier coefficients?

Since the vectors $\{\mathbf{z}^0,\dots,\mathbf{z}^{n-1}\}$ are orthonormal, we can compute the k-th Fourier coefficient of $\mathbf{g}$ by computing the dot-product:

$$\begin{aligned}\hat{\mathbf{g}}_k &= \langle \mathbf{g}, \mathbf{z}^k\rangle_{[0,2\pi)}\\ &= \frac{2\pi}{n}\cdot\sum_{j=0}^{n-1}\mathbf{g}_j\cdot\bar{\mathbf{z}}_j^k\\ &= \frac{\sqrt{2\pi}}{n}\cdot\sum_{j=0}^{n-1}\mathbf{g}_j\cdot e^{-\frac{ik2\pi j}{n}}\end{aligned}$$

Challenge

How can we compute all n Fourier coefficients?

Since the vectors $\{\mathbf{z}^0, \ldots, \mathbf{z}^{n-1}\}$ are orthonormal, we can compute the k-th Fourier coefficient of $\mathbf{g}$ by computing the dot-product:

$$\hat{\mathbf{g}}_k = \langle \mathbf{g}, \mathbf{z}^k \rangle_{[0,2\pi)}$$

Computing one coefficient has complexity $O(n)$

Computing all of them has complexity $O(n^2)$

πj n $\sum_{j=0}$ $\mathbf{g}_j$

Challenge

How can we compute all n Fourier coefficients efficiently?

Approach (Divide and Conquer)

Key Idea:

If we decompose the array into the even and odd halves, we can solve smaller problems and then combine.

Approach (Divide and Conquer)

Key Idea:

Consider the 8-dimensional array:

$$\mathbf{g} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4, \mathbf{g}_5, \mathbf{g}_6, \mathbf{g}_7)$$

And consider its even/odd decomposition:

$$\begin{aligned}\mathbf{g}^e &= (\mathbf{g}_0, \mathbf{g}_2, \mathbf{g}_4, \mathbf{g}_6)\\ \mathbf{g}^o &= (\mathbf{g}_1, \mathbf{g}_3, \mathbf{g}_5, \mathbf{g}_7)\end{aligned}$$

Consider how the coefficients of $\mathbf{g}$ are weighted when computing the k-th Fourier coefficient, relative to how the coefficients of $\mathbf{g}^e$ and $\mathbf{g}^o$ are weighted.

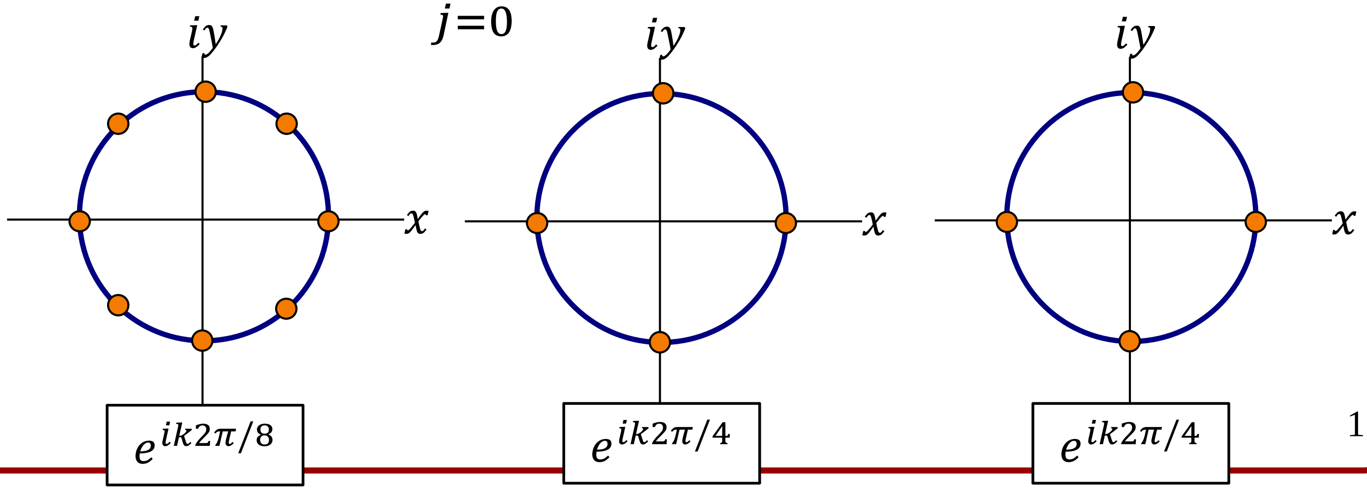
Approach (Divide and Conquer)

Key Idea:

$$\mathbf{g} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4, \mathbf{g}_5, \mathbf{g}_6, \mathbf{g}_7)$$
$$\mathbf{g}^e = (\mathbf{g}_0, \mathbf{g}_2, \mathbf{g}_4, \mathbf{g}_6)$$
$$\mathbf{g}^o = (\mathbf{g}_1, \mathbf{g}_3, \mathbf{g}_5, \mathbf{g}_7)$$

Now let's consider the Fourier coefficients:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{n-1} \mathbf{g}_j \cdot e^{-\frac{ik2\pi j}{n}}$$



Approach (Divide and Conquer)

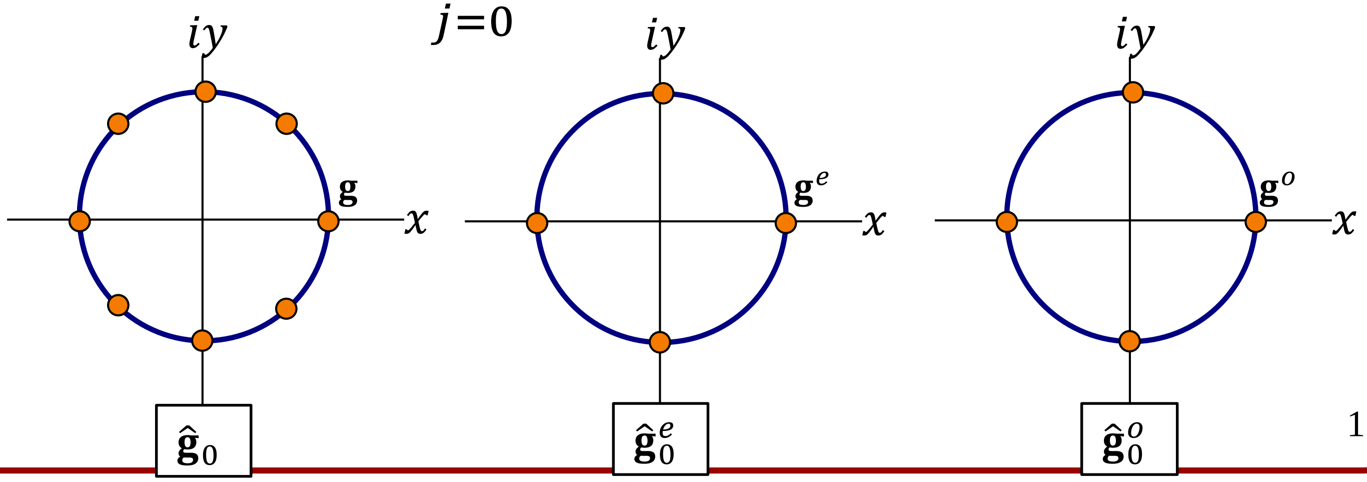
Key Idea:

$$\mathbf{g} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4, \mathbf{g}_5, \mathbf{g}_6, \mathbf{g}_7)$$
$$\mathbf{g}^e = (\mathbf{g}_0, \mathbf{g}_2, \mathbf{g}_4, \mathbf{g}_6)$$
$$\mathbf{g}^o = (\mathbf{g}_1, \mathbf{g}_3, \mathbf{g}_5, \mathbf{g}_7)$$

Now let's consider the Fourier coefficients:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{n-1} \mathbf{g}_j \cdot e^{-\frac{ik2\pi j}{n}}$$

$\boxed{k = 0}$



Approach (Divide and Conquer)

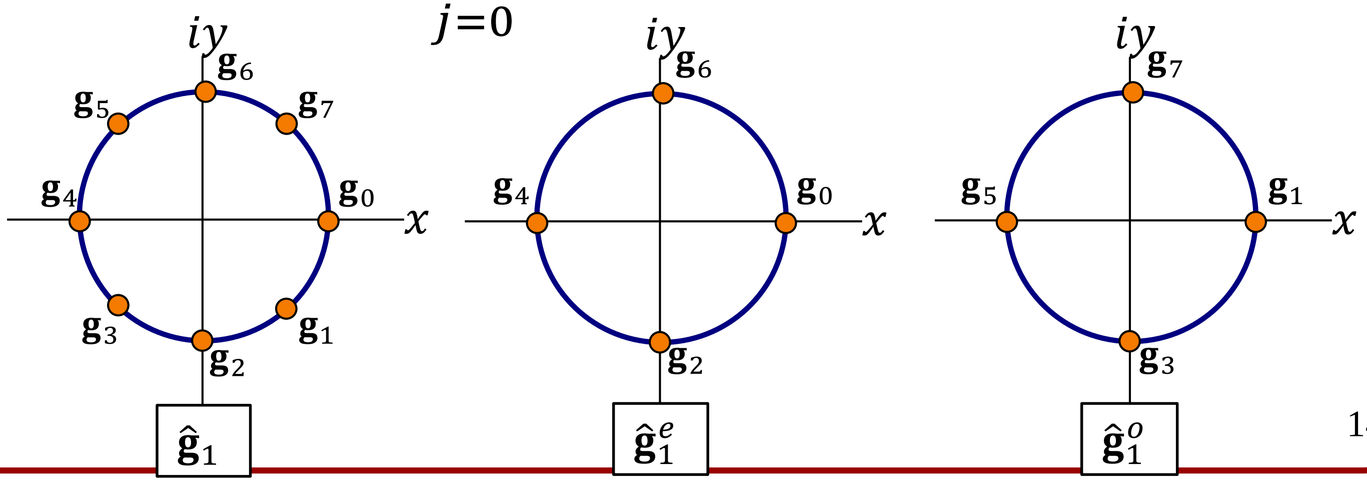
Key Idea:

$$\mathbf{g} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4, \mathbf{g}_5, \mathbf{g}_6, \mathbf{g}_7)$$
$$\mathbf{g}^e = (\mathbf{g}_0, \mathbf{g}_2, \mathbf{g}_4, \mathbf{g}_6)$$
$$\mathbf{g}^o = (\mathbf{g}_1, \mathbf{g}_3, \mathbf{g}_5, \mathbf{g}_7)$$

Now let's consider the Fourier coefficients:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{n-1} \mathbf{g}_j \cdot e^{-\frac{ik2\pi j}{n}}$$

$$\boxed{k = 1}$$



Approach (Divide and Conquer)

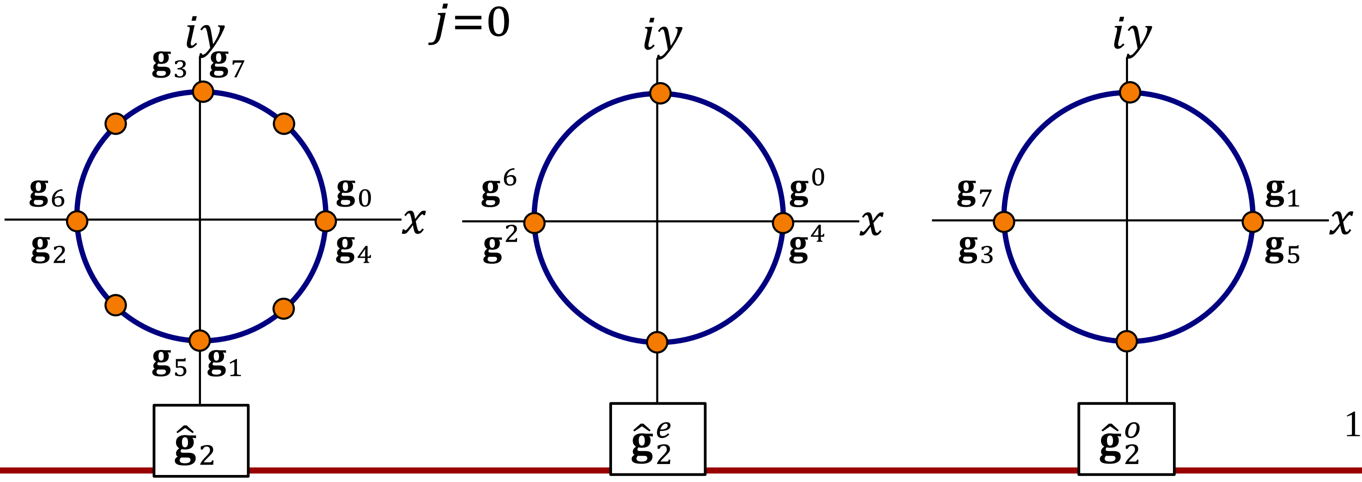
Key Idea:

$$\mathbf{g} = (\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4, \mathbf{g}_5, \mathbf{g}_6, \mathbf{g}_7)$$
$$\mathbf{g}^e = (\mathbf{g}_0, \mathbf{g}_2, \mathbf{g}_4, \mathbf{g}_6)$$
$$\mathbf{g}^o = (\mathbf{g}_1, \mathbf{g}_3, \mathbf{g}_5, \mathbf{g}_7)$$

Now let's consider the Fourier coefficients:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{n-1} \mathbf{g}_j \cdot e^{-\frac{ik2\pi j}{n}}$$

$\boxed{k = 2}$



Approach (Divide and Conquer)

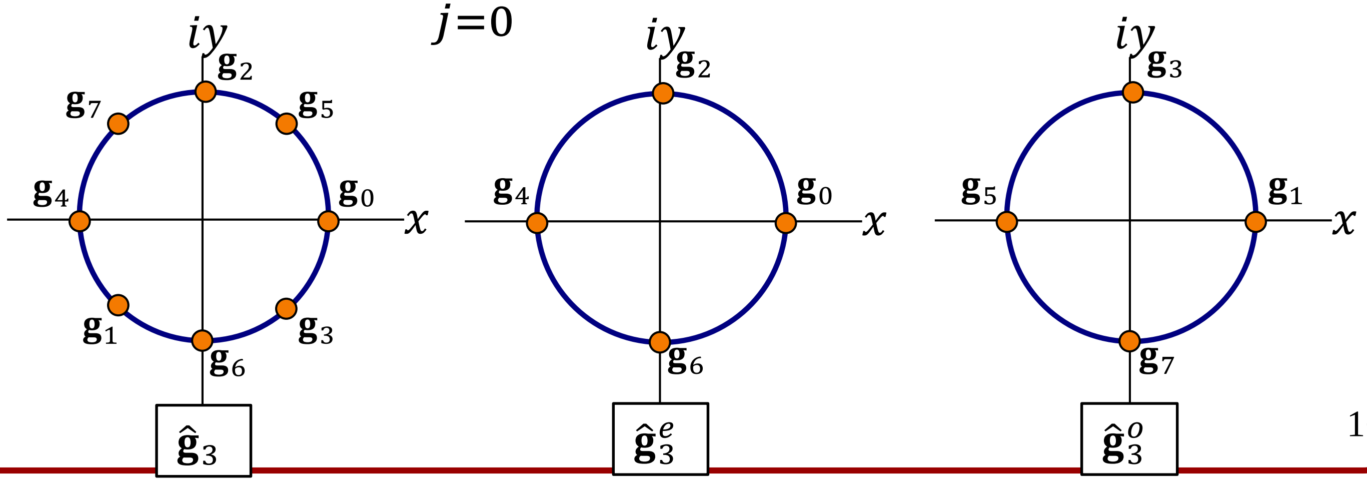
Key Idea:

$$\begin{aligned}\mathbf{g} &= (\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4, \mathbf{g}_5, \mathbf{g}_6, \mathbf{g}_7)\\ \mathbf{g}^e &= (\mathbf{g}_0, \mathbf{g}_2, \mathbf{g}_4, \mathbf{g}_6)\\ \mathbf{g}^o &= (\mathbf{g}_1, \mathbf{g}_3, \mathbf{g}_5, \mathbf{g}_7)\end{aligned}$$

Now let's consider the Fourier coefficients:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{n-1} \mathbf{g}_j \cdot e^{-\frac{ik2\pi j}{n}}$$

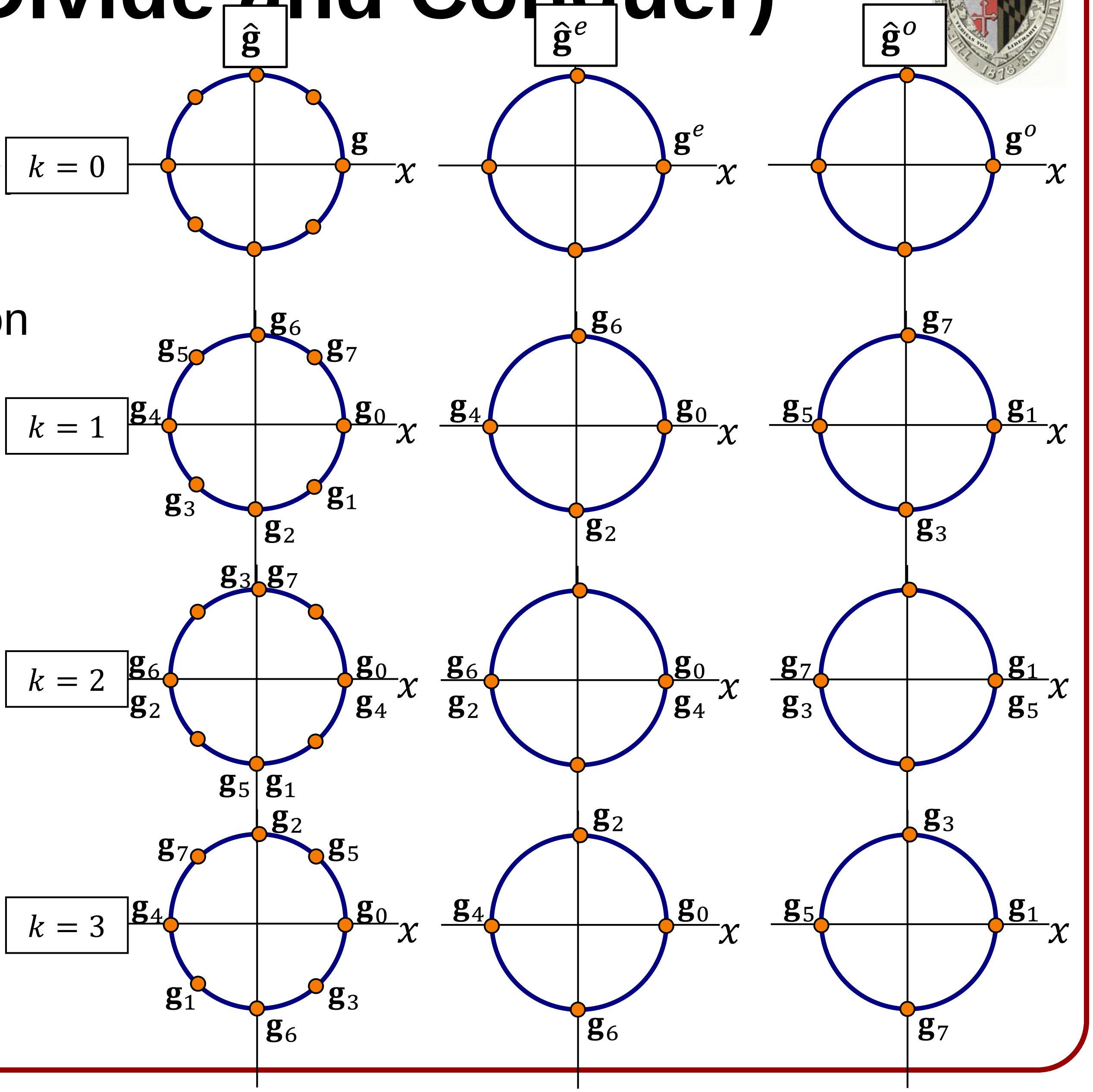
$\boxed{k = 3}$



Approach (Divide and Conquer)

For every frequency:

- The even components are in the same position as the original
- The odd ones are off by a fixed angle.



$\hat{\mathbf{g}}$
$\hat{\mathbf{g}}^e$
$\hat{\mathbf{g}}^o$
$k = 0$
$\mathbf{g}$
$\mathbf{g}^e$
$\mathbf{g}^o$
x
$k = 1$
$\mathbf{g}_6$ $\mathbf{g}_5$ $\mathbf{g}_7$ $\mathbf{g}_4$ $\mathbf{g}_0$ $\mathbf{g}_3$ $\mathbf{g}_1$ $\mathbf{g}_2$
$\mathbf{g}_6$ $\mathbf{g}_4$ $\mathbf{g}_0$ $\mathbf{g}_2$
$\mathbf{g}_7$ $\mathbf{g}_5$ $\mathbf{g}_1$ $\mathbf{g}_3$
$k = 2$
$\mathbf{g}_3$ $\mathbf{g}_7$ $\mathbf{g}_6$ $\mathbf{g}_2$ $\mathbf{g}_0$ $\mathbf{g}_4$ $\mathbf{g}_5$ $\mathbf{g}_1$
$\mathbf{g}_6$ $\mathbf{g}_2$ $\mathbf{g}_0$ $\mathbf{g}_4$
$\mathbf{g}_7$ $\mathbf{g}_3$ $\mathbf{g}_1$ $\mathbf{g}_5$
$k = 3$
$\mathbf{g}_2$ $\mathbf{g}_7$ $\mathbf{g}_5$ $\mathbf{g}_4$ $\mathbf{g}_0$ $\mathbf{g}_1$ $\mathbf{g}_3$ $\mathbf{g}_6$
$\mathbf{g}_2$ $\mathbf{g}_4$ $\mathbf{g}_0$ $\mathbf{g}_6$
$\mathbf{g}_3$ $\mathbf{g}_5$ $\mathbf{g}_1$ $\mathbf{g}_7$

Approach (Divide and Conquer)

Assume that n is even ($n = 2m$) and lets consider the even and odd entries separately.

That is, let $\mathbf{g}^e, \mathbf{g}^o \in \mathbb{C}^m$ be the (periodic) arrays:

$$\mathbf{g}^e_k = \mathbf{g}_{2k} \qquad \mathbf{g}^o_k = \mathbf{g}_{2k+1}$$

Approach (Divide and Conquer)

$$\mathbf{g}_k^e = \mathbf{g}_{2k} \qquad \mathbf{g}_k^o = \mathbf{g}_{2k+1}$$

The k-th Fourier coefficient of $\mathbf{g}$ is:

$$\hat{\mathbf{g}}_k = \frac{1}{\sqrt{2\pi}} \cdot \sum_{j=0}^{n-1} \mathbf{g}_j \cdot e^{-\frac{ik2\pi j}{n}}$$

Splitting the summation into even/odd terms:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{m-1} \left(\mathbf{g}_{2j} \cdot e^{-\frac{ik2\pi(2j)}{2m}} + \mathbf{g}_{2j+1} \cdot e^{-\frac{ik2\pi(2j+1)}{2m}} \right)$$

Approach (Divide and Conquer)

$$\mathbf{g}_k^e = \mathbf{g}_{2k} \qquad \mathbf{g}_k^o = \mathbf{g}_{2k+1}$$

Splitting the summation into even/odd terms:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{m-1} \left(\mathbf{g}_{2j} \cdot e^{-\frac{ik2\pi(2j)}{2m}} + \mathbf{g}_{2j+1} \cdot e^{-\frac{ik2\pi(2j+1)}{2m}} \right)$$

Re-writing the exponent, we get:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{m-1} \left(\mathbf{g}_{2j} \cdot e^{-\frac{ik2\pi j}{m}} + \mathbf{g}_{2j+1} \cdot e^{-\frac{ik2\pi j}{m}} \cdot e^{-\frac{ik2\pi}{n}} \right)$$

Approach (Divide and Conquer)

$$\mathbf{g}^e_k = \mathbf{g}_{2k} \qquad \mathbf{g}^o_k = \mathbf{g}_{2k+1}$$

Re-writing the exponent, we get:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{m-1} \left(\mathbf{g}_{2j} \cdot e^{-\frac{ik2\pi j}{m}} + \mathbf{g}_{2j+1} \cdot e^{-\frac{ik2\pi j}{m}} \cdot e^{-\frac{ik2\pi}{n}} \right)$$

Plugging in our even/odd expression gives:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{m-1} \left(\mathbf{g}^e_j \cdot e^{-\frac{ik2\pi j}{m}} + \mathbf{g}^o_j \cdot e^{-\frac{ik2\pi j}{m}} \cdot e^{-\frac{ik2\pi}{n}} \right)$$

Approach (Divide and Conquer)

$$\mathbf{g}_k^e = \mathbf{g}_{2k} \qquad \mathbf{g}_k^o = \mathbf{g}_{2k+1}$$

Plugging in our even/odd expression gives:

$$\hat{\mathbf{g}}_k = \frac{\sqrt{2\pi}}{n} \cdot \sum_{j=0}^{m-1} \left(\mathbf{g}_j^e \cdot e^{-\frac{ik2\pi j}{m}} + \mathbf{g}_j^o \cdot e^{-\frac{ik2\pi j}{m}} \cdot e^{-\frac{ik2\pi}{n}} \right)$$

Re-writing this equation gives:

$$\hat{\mathbf{g}}_k = \left(\frac{1}{2} \cdot \frac{\sqrt{2\pi}}{m} \cdot \sum_{j=0}^{m-1} \mathbf{g}_j^e \cdot e^{-\frac{ik2\pi j}{m}} \right) + e^{-\frac{ik2\pi}{n}} \cdot \left(\frac{1}{2} \cdot \frac{\sqrt{2\pi}}{m} \cdot \sum_{j=0}^{m-1} \mathbf{g}_j^o \cdot e^{-\frac{ik2\pi j}{m}} \right)$$

Approach (Divide and Conquer)

$$\mathbf{g}^e_k = \mathbf{g}_{2k} \qquad \mathbf{g}^o_k = \mathbf{g}_{2k+1}$$

Re-writing this equation gives:

$$\hat{\mathbf{g}}_k = \left(\frac{1}{2}\cdot\frac{\sqrt{2\pi}}{m}\cdot\sum_{j=0}^{m-1}\mathbf{g}^e_j\cdot e^{-\frac{ik2\pi j}{m}}\right) + e^{-\frac{ik2\pi}{n}}\cdot\left(\frac{1}{2}\cdot\frac{\sqrt{2\pi}}{m}\cdot\sum_{j=0}^{m-1}\mathbf{g}^o_j\cdot e^{-\frac{ik2\pi j}{m}}\right)$$

But this is a weighted sum of Fourier coefficients:

$$\hat{\mathbf{g}}_k = \frac{1}{2}\cdot\left(\hat{\mathbf{g}}^e_k + \hat{\mathbf{g}}^o_k\cdot e^{-\frac{ik2\pi}{n}}\right)$$

Approach (Divide and Conquer)

$$\mathbf{g}_k^e = \mathbf{g}_{2k} \qquad \mathbf{g}_k^o = \mathbf{g}_{2k+1}$$

$$\hat{\mathbf{g}}_k = \frac{1}{2} \cdot \left(\hat{\mathbf{g}}_k^e + \hat{\mathbf{g}}_k^o \cdot e^{-\frac{ik2\pi}{n}} \right)$$

If we get the Fourier coefficients of $\mathbf{g}^e$ and $\mathbf{g}^o$, we can combine them to get the Fourier coefficients of $\mathbf{g}$.

Assuming that m is also even, we can repeat for $\mathbf{g}^e$ and $\mathbf{g}^o$ to get their Fourier coefficients.

If n is a power of 2, we can keep splitting, to get an $O(n \log n)$ algorithm.

The Inverse Fourier Transform

It turns out that computing the inverse Fourier transform is (almost) equivalent to computing the forward transform…

The Inverse Fourier Transform

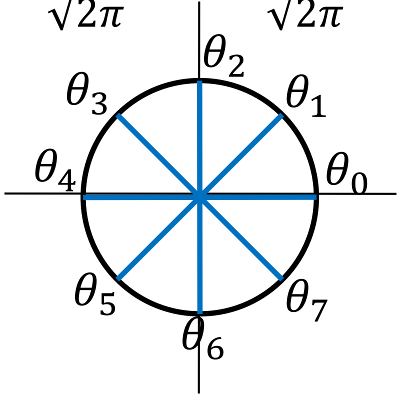
The Fourier Transform is a change of basis:

Evaluation Basis

$$\begin{matrix}(1,0,\cdots,0,0)\\(0,1,\cdots,0,0)\\\vdots\\(0,0,\cdots,1,0)\\(0,0,\cdots,0,1)\end{matrix}\xrightarrow{\text{Fourier Transform}}$$

Complex Exponential Basis

$$\left(\frac{e^{i0\theta_0}}{\sqrt{2\pi}},\frac{e^{i0\theta_1}}{\sqrt{2\pi}},\dots,\frac{e^{i0\theta_{n-2}}}{\sqrt{2\pi}},\frac{e^{i0\theta_{n-1}}}{\sqrt{2\pi}}\right)$$
$$\left(\frac{e^{i1\theta_0}}{\sqrt{2\pi}},\frac{e^{i1\theta_1}}{\sqrt{2\pi}},\dots,\frac{e^{i1\theta_{n-2}}}{\sqrt{2\pi}},\frac{e^{i1\theta_{n-1}}}{\sqrt{2\pi}}\right)$$
$$\vdots$$
$$\left(\frac{e^{i(n-2)\theta_0}}{\sqrt{2\pi}},\frac{e^{i(n-2)\theta_1}}{\sqrt{2\pi}},\dots,\frac{e^{i(n-2)\theta_{n-2}}}{\sqrt{2\pi}},\frac{e^{i(n-2)\theta_{n-1}}}{\sqrt{2\pi}}\right)$$
$$\left(\frac{e^{i(n-1)\theta_0}}{\sqrt{2\pi}},\frac{e^{i(n-1)\theta_1}}{\sqrt{2\pi}},\dots,\frac{e^{i(n-1)\theta_{n-2}}}{\sqrt{2\pi}},\frac{e^{i(n-1)\theta_{n-1}}}{\sqrt{2\pi}}\right)$$



The Inverse Fourier Transform

The Fourier Transform is a change of basis:

Complex Exponential Basis

$$\left(\frac{e^{i0\theta_0}}{\sqrt{2\pi}},\frac{e^{i0\theta_1}}{\sqrt{2\pi}},\cdots,\frac{e^{i0\theta_{n-2}}}{\sqrt{2\pi}},\frac{e^{i0\theta_{n-1}}}{\sqrt{2\pi}}\right)$$
$$\left(\frac{e^{i1\theta_0}}{\sqrt{2\pi}},\frac{e^{i1\theta_1}}{\sqrt{2\pi}},\cdots,\frac{e^{i1\theta_{n-2}}}{\sqrt{2\pi}},\frac{e^{i1\theta_{n-1}}}{\sqrt{2\pi}}\right)$$
$$\vdots$$
$$\frac{{}^{i(n-2)\theta_{n-1}}}{\sqrt{2\pi}}\Bigg)$$

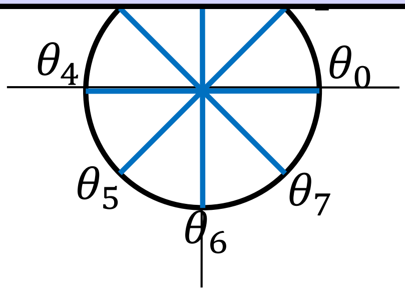
Evaluation Basis

$$(1,0,\cdots,0,0)$$
$$(0,1,\cdots,0,0)$$
$$\vdots$$
$$(0,0,\cdots,1,0)$$
$$(0,0,\cdots,0,1)$$

$$\xrightarrow{\text{Fourier Transform}}$$

<u>Note</u>:
The evaluation basis is <u>not</u> orthonormal.

It can be made orthonormal by scaling all the basis vectors by the same scale factor, $\sqrt{n}/\sqrt{2\pi}$.



The Inverse Fourier Transform

Since the Fourier basis $\{\mathbf{z}^0, \ldots, \mathbf{z}^{n-1}\}$ is orthonormal, the k-th Fourier coefficient of a vector $\mathbf{a} \in \mathbb{C}^n$ is the (Hermitian) dot-product:

$$\begin{aligned}\hat{\mathbf{a}}_k &= \langle \mathbf{a}, \mathbf{z}^k \rangle \\ &= \frac{2\pi}{n}\left(\bar{\mathbf{z}}_0^k, \ldots, \bar{\mathbf{z}}_{n-1}^k\right) \cdot \begin{pmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_{n-1} \end{pmatrix} \\ &= \frac{2\pi}{n}\left(\frac{e^{-ik\cdot 0\theta}}{\sqrt{2\pi}}, \ldots, \frac{e^{-ik\cdot(n-1)\theta}}{\sqrt{2\pi}}\right) \cdot \begin{pmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_{n-1} \end{pmatrix}\end{aligned}$$

where θ is the angle $\theta = \frac{2\pi}{n}$.

The Inverse Fourier Transform

$$\hat{\mathbf{a}}_k = \frac{\sqrt{2\pi}}{n}\left(e^{-ik\cdot 0\theta},\dots,e^{-ik\cdot(n-1)\theta}\right)\cdot\begin{pmatrix}\mathbf{a}_0\\ \vdots\\ \mathbf{a}_{n-1}\end{pmatrix}, \qquad \text{w/ } \theta = \frac{2\pi}{n}$$

Stacking the rows we can represent the Fourier transform by the matrix:

$$\mathbf{F} = \frac{\sqrt{2\pi}}{n}\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & e^{-i\theta} & \cdots & e^{-i(n-2)\theta} & e^{-i(n-1)\theta} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-i(n-2)\theta} & \cdots & e^{-i(n-2)(n-2)\theta} & e^{-i(n-2)(n-1)\theta} \\ 1 & e^{-i(n-1)\theta} & \cdots & e^{-i(n-1)(n-2)\theta} & e^{-i(n-1)(n-1)\theta} \end{pmatrix}.$$

The Inverse Fourier Transform

$$\mathbf{F} = \frac{\sqrt{2\pi}}{n}\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & e^{-i\theta} & \cdots & e^{-i(n-2)\theta} & e^{-i(n-1)\theta} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-i(n-2)\theta} & \cdots & e^{-i(n-2)(n-2)\theta} & e^{-i(n-2)(n-1)\theta} \\ 1 & e^{-i(n-1)\theta} & \cdots & e^{-i(n-1)(n-2)\theta} & e^{-i(n-1)(n-1)\theta} \end{pmatrix}$$

Since the evaluation and Fourier bases are (essentially) orthonormal[*], the matrix is unitary, and the inverse Fourier transform is the transpose conjugate of the forward transform.

[*]Up to a scale factor of $\sqrt{n/2\pi}$ the evaluation basis is orthonormal.

The Inverse Fourier Transform

$$\mathbf{F} = \frac{\sqrt{2\pi}}{n}\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & e^{-i\theta} & \cdots & e^{-i(n-2)\theta} & e^{-i(n-1)\theta} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-i(n-2)\theta} & \cdots & e^{-i(n-2)(n-2)\theta} & e^{-i(n-2)(n-1)\theta} \\ 1 & e^{-i(n-1)\theta} & \cdots & e^{-i(n-1)(n-2)\theta} & e^{-i(n-1)(n-1)\theta} \end{pmatrix}$$

In particular, given the Fourier coefficients:

$$\hat{\mathbf{a}} = (\hat{\mathbf{a}}_0, \cdots, \hat{\mathbf{a}}_{n-1})$$

the inverse Fourier transform is:

$$\mathbf{F}^{-1}\hat{\mathbf{a}} = c \cdot \bar{\mathbf{F}}^t\hat{\mathbf{a}}$$

(for some constant c accounting for the non-orthonormality of the evaluation basis).

The Inverse Fourier Transform

$$\mathbf{F} = \frac{\sqrt{2\pi}}{n}\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & e^{-i\theta} & \cdots & e^{-i(n-2)\theta} & e^{-i(n-1)\theta} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-i(n-2)\theta} & \cdots & e^{-i(n-2)(n-2)\theta} & e^{-i(n-2)(n-1)\theta} \\ 1 & e^{-i(n-1)\theta} & \cdots & e^{-i(n-1)(n-2)\theta} & e^{-i(n-1)(n-1)\theta} \end{pmatrix}$$

Taking the double conjugate, we get:

$$\begin{aligned} \mathbf{F}^{-1}\hat{\mathbf{a}} &= c \cdot \bar{\mathbf{F}}^t\hat{\mathbf{a}} \\ &= c \cdot \overline{\overline{\bar{\mathbf{F}}^t\hat{\mathbf{a}}}} \\ &= c \cdot \overline{\mathbf{F}^t\bar{\hat{\mathbf{a}}}} \end{aligned}$$

The Inverse Fourier Transform

$$\mathbf{F} = \frac{\sqrt{2\pi}}{n}\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & e^{-i\theta} & \cdots & e^{-i(n-2)\theta} & e^{-i(n-1)\theta} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-i(n-2)\theta} & \cdots & e^{-i(n-2)(n-2)\theta} & e^{-i(n-2)(n-1)\theta} \\ 1 & e^{-i(n-1)\theta} & \cdots & e^{-i(n-1)(n-2)\theta} & e^{-i(n-1)(n-1)\theta} \end{pmatrix}$$

Since $\mathbf{F} = \mathbf{F}^t$:

$$\begin{aligned} \mathbf{F}^{-1}\hat{\mathbf{a}} &= c \cdot \overline{\mathbf{F}^t \overline{\hat{\mathbf{a}}}} \\ &= c \cdot \overline{\mathbf{F}\overline{\hat{\mathbf{a}}}} \end{aligned}$$

The Inverse Fourier Transform

$$\mathbf{F}^{-1}\hat{\mathbf{a}} = c\cdot\overline{\mathbf{F}\overline{\hat{\mathbf{a}}}}$$

Q: What is the constant c?

A: The matrix $\mathbf{F}$, gives the Fourier coefficients of a vector represented in the evaluation basis.

⇒ The evaluation basis becomes orthonormal if we scale by $\sqrt{n/2\pi}$.

⇒ The matrix $\sqrt{2\pi/n}\cdot\mathbf{F}$ is a change of basis between two orthonormal bases.

⇒ The matrix $\sqrt{2\pi/n}\cdot\mathbf{F}$ is unitary.

The Inverse Fourier Transform

$$\mathbf{F}^{-1}\hat{\mathbf{a}} = c \cdot \overline{\mathbf{F}\overline{\hat{\mathbf{a}}}}$$

$\Rightarrow$ The matrix $\sqrt{2\pi/n} \cdot \mathbf{F}$ is unitary.

$\Rightarrow$ Its inverse is its conjugate transpose:

$$\left(\sqrt{2\pi/n} \cdot \mathbf{F}\right)^{-1} = \sqrt{2\pi/n} \cdot \bar{\mathbf{F}}^{t}$$

$$\Updownarrow$$

$$\mathbf{F}^{-1} = 2\pi/n \cdot \bar{\mathbf{F}}^{t}$$

$\Rightarrow$ The inverse Fourier transform is defined by:

$$\mathbf{F}^{-1}\hat{\mathbf{a}} = 2\pi/n \cdot \overline{\mathbf{F}\overline{\hat{\mathbf{a}}}}$$

The Inverse Fourier Transform

$$\mathbf{F}^{-1}\hat{\mathbf{a}} = 2\pi/n \cdot \overline{\mathbf{F}\overline{\hat{\mathbf{a}}}}$$

Normalized Inverse Fourier Transform:

1. Take the conjugate of the Fourier coefficients
2. Compute the forward Fourier transform
3. Take the conjugate of the resultant coefficients
4. Scale by $2\pi/n$

The Inverse Fourier Transform

$$\mathbf{F}^{-1}\hat{\mathbf{a}} = 2\pi/n \cdot \overline{\mathbf{F}\bar{\hat{\mathbf{a}}}}$$

Note:

To avoid an expensive square root, most FFT implementations compute the un-normalized Fourier coefficients:

$$\tilde{\hat{\mathbf{a}}}_k = \left(\bar{\mathbf{z}}_0^k, \dots, \bar{\mathbf{z}}_{n-1}^k\right) \cdot \begin{pmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_{n-1} \end{pmatrix} = \sqrt{2\pi}\hat{\mathbf{a}}_k$$

$\Rightarrow$ The two Fourier transforms are related by:

$$\tilde{\mathbf{F}} = \sqrt{2\pi}\mathbf{F}$$

The Inverse Fourier Transform

$$\mathbf{F}^{-1}\hat{\mathbf{a}} = 2\pi/n \cdot \overline{\mathbf{F}\overline{\hat{\mathbf{a}}}}$$
$$\tilde{\mathbf{F}} = \sqrt{2\pi}\mathbf{F}$$

<u>Note</u>:

Taking the inverse, we get:

$$\begin{aligned}
\tilde{\mathbf{F}}^{-1}\tilde{\hat{\mathbf{a}}} &= 1/\sqrt{2\pi} \cdot \mathbf{F}^{-1}\tilde{\hat{\mathbf{a}}} \\
&= 1/\sqrt{2\pi} \cdot 2\pi/n \cdot \overline{\mathbf{F}\overline{\tilde{\hat{\mathbf{a}}}}} \\
&= 1/\sqrt{2\pi} \cdot 2\pi/n \cdot 1/\sqrt{2\pi} \cdot \overline{\tilde{\mathbf{F}}\overline{\tilde{\hat{\mathbf{a}}}}} \\
&= 1/n \cdot \overline{\tilde{\mathbf{F}}\overline{\tilde{\hat{\mathbf{a}}}}}
\end{aligned}$$

The Inverse Fourier Transform

$$\tilde{\mathbf{F}}^{-1}\hat{\tilde{\mathbf{a}}} = 1/n \cdot \overline{\tilde{\mathbf{F}}\overline{\hat{\tilde{\mathbf{a}}}}}$$

Unnormalized Inverse Fourier Transform:

1. Take the conjugate of the Fourier coefficients
2. Compute the forward Fourier transform
3. Take the conjugate of the resultant coefficients
4. Scale by $1/n$

Implementations like the FFTW do not adjust for the scaling term.

$\Downarrow$

Running the forward Fourier Transform and then the inverse scales the input by a factor of $1/n$.

Outline

The FFT Algorithm

Multi-Dimensional FFTs

More Applications

Real FFTs

Multi-Dimensional FFTs

How do we compute the Fourier transform of a multi-dimensional signal?

Examples:

- Images
- Voxel Grids
- Etc.

2D FFTs

For regularly sampled, $n \times n$ grids, the irreducible representations are spanned by the orthonormal basis $\{\mathbf{z}^{lm}\}$ where:

$$\mathbf{z}^{lm}_{jk} = \frac{1}{2\pi} \cdot e^{\frac{il2\pi j}{n}} \cdot e^{\frac{im2\pi k}{n}}$$

To see this, consider the shift by (α, β):

$$\begin{aligned}\left(\rho_{\alpha,\beta}\left(\mathbf{z}^{lm}\right)\right)_{jk} &= \mathbf{z}^{lm}_{j-\alpha,k-\beta} \\ &= \frac{1}{2\pi} \cdot e^{\frac{il2\pi(j-\alpha)}{n}} \cdot e^{\frac{im2\pi(k-\beta)}{n}} \\ &= \left(e^{-\frac{il2\pi\alpha}{n}} \cdot e^{-\frac{im2\pi\beta}{n}}\right) \cdot \mathbf{z}^{lm}_{jk}\end{aligned}$$

2D FFTs

How can we compute the 2D Fourier coefficients efficiently?

To do this, we will leverage the fact that the 2D basis vectors can be expressed as the product of 1D basis vectors.

Setting $\mathbf{z}^l$ to be the n-dimensional array:

$$\mathbf{z}_j^l = \frac{1}{\sqrt{2\pi}} \cdot e^{\frac{il2\pi j}{n}}$$

we get:

$$\mathbf{z}_{jk}^{lm} = \mathbf{z}_j^l \cdot \mathbf{z}_k^m$$

2D FFTs

To compute the (l,m)-th Fourier coefficient of an $(n \times n)$-dimensional grid $\mathbf{g}$, we compute the dot-product of $\mathbf{g}$ with $\mathbf{z}^{lm}$:

$$\begin{aligned}
\hat{\mathbf{g}}_{lm} &= \langle \mathbf{g}, \mathbf{z}^{lm} \rangle_{[0,2\pi)\times[0,2\pi)} \\
&= \frac{(2\pi)^2}{n^2} \cdot \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} \mathbf{g}_{jk} \cdot \bar{\mathbf{z}}_{jk}^{lm} \\
&= \frac{(2\pi)^2}{n^2} \cdot \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} \mathbf{g}_{jk} \cdot \bar{\mathbf{z}}_{j}^{l} \cdot \bar{\mathbf{z}}_{k}^{m} \\
&= \frac{2\pi}{n} \cdot \sum_{j=0}^{n-1} \left(\frac{2\pi}{n} \cdot \sum_{k=0}^{n-1} \mathbf{g}_{jk} \cdot \bar{\mathbf{z}}_{j}^{l} \right) \cdot \bar{\mathbf{z}}_{k}^{m}
\end{aligned}$$

2D FFTs

$$\hat{\mathbf{g}}_{lm} = \frac{2\pi}{n} \cdot \sum_{j=0}^{n-1} \boxed{\left(\frac{2\pi}{n} \cdot \sum_{k=0}^{n-1} \mathbf{g}_{jk} \cdot \bar{\mathbf{z}}_j^l \right)} \cdot \bar{\mathbf{z}}_k^m$$

The interior sum is a 1D Fourier transform!

In particular, setting $\mathbf{g}^j$ to be the 1D array:

$$\mathbf{g}_k^j = \mathbf{g}_{jk}$$

we get:

$$\begin{aligned} \hat{\mathbf{g}}_{lm} &= \frac{2\pi}{n} \cdot \sum_{j=0}^{n-1} \left(\frac{2\pi}{n} \cdot \sum_{k=0}^{n-1} \mathbf{g}_k^j \cdot \bar{\mathbf{z}}_j^l \right) \cdot \bar{\mathbf{z}}_k^m \\ &= \frac{2\pi}{n} \cdot \sum_{j=0}^{n-1} \hat{\mathbf{g}}_m^j \cdot \bar{\mathbf{z}}_k^m \end{aligned}$$

2D FFTs

$$\hat{\mathbf{g}}_{lm} = \frac{2\pi}{n} \cdot \sum_{j=0}^{n-1} \hat{\mathbf{g}}_m^j \cdot \bar{\mathbf{z}}_k^m$$

But this sum is also a 1D Fourier transform!

Thus, to compute the Fourier coefficients, we:

1. Compute the Fourier coefficients of each row
2. Compute the Fourier coefficients of each column

And the total complexity of this operation is:

$$O(n^2 \log n)$$

Outline

The FFT Algorithm

Multi-Dimensional FFTs

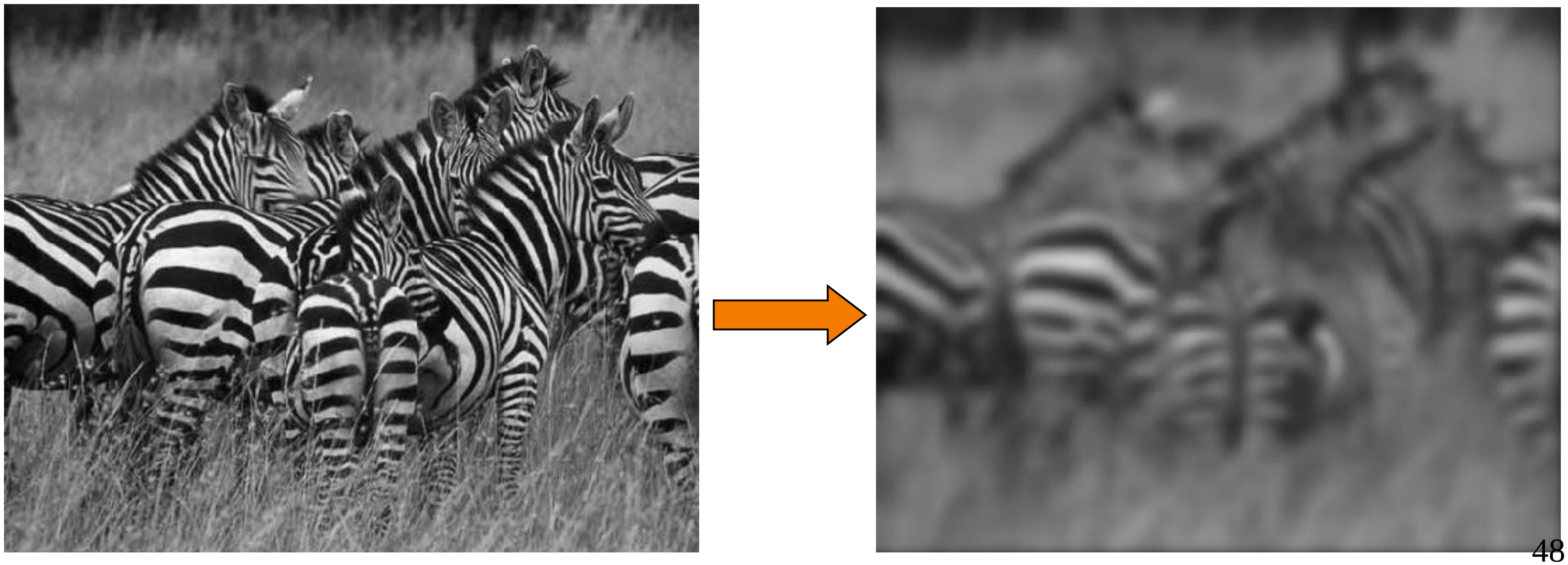
More Applications

- Gaussian Smoothing
- Up-Sampling
- Differentiation
- Boundary Detection
- Gaussian Sharpening

Real FFTs

Gaussian Smoothing

Given an $n \times n$ grid of values, we would like to smooth the grid.



Gaussian Smoothing

Given an $n \times n$ grid of values, we would like to smooth the grid.

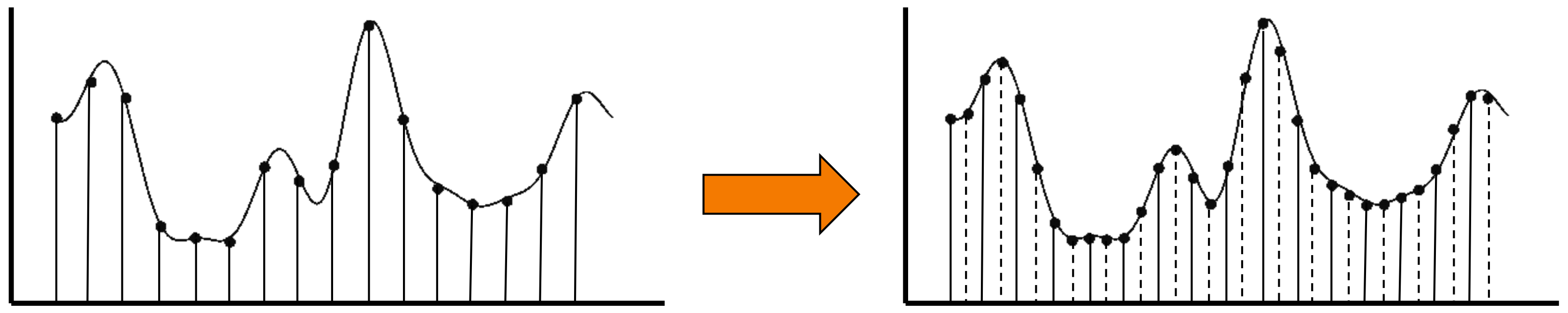
To do this we need:

- $\mathbf{f}$: The smoothing filter, usually a Gaussian:
  $$\tilde{\mathbf{f}}_{jk} = e^{-(j^2+k^2)/2\sigma^2} \quad \text{with} \quad -\frac{n}{2} < j,k \leq \frac{n}{2}$$
  normalized to have unit mass:
  $$\mathbf{f}_{jk} = \frac{\tilde{\mathbf{f}}_{jk}}{\langle \tilde{\mathbf{f}}, 1 \rangle_{[0,2\pi)\times[0,2\pi)}}$$
- $\mathbf{g}$: The initial grid
- $\mathbf{f} * \mathbf{g}$: The Gaussian-smoothed grid

Up-Sampling

Given an n-dimensional array, we would like to extrapolate the array to a $2n$-dimensional array.

One way to do this is to fit a continuous function to the original array and then sample at $2n$ regular samples.



Up-Sampling

How do we generate a continuous function from a set of n samples?

Recall that the Fourier decomposition of $\mathbf{g}$ is:

$$\mathbf{g}_j = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{\frac{ik2\pi j}{n}}$$

Up-Sampling

How do we generate a continuous function from a set of n samples?

We can fit a continuous function to the data by associating the discrete index $0 \leq j \leq n$ with a continuous index $0 \leq s \leq 2\pi$:

$$\mathbf{g}_j = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{\frac{ik2\pi j}{n}} \Rightarrow g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{iks}$$

Since at integer values of j we have:

$$\mathbf{g}_j = g(2\pi j/n)$$

the continuous function interpolates the n discrete samples.

Up-Sampling

<u>Word of Warning</u>:

Recall that for integer values of j, the complex exponential satisfies the condition:

$$e^{\frac{ik2\pi j}{n}} = e^{\frac{i(k+n)2\pi j}{n}}$$

Thus, if we were to fit a function:

$$\mathbf{g}_j = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{\frac{ik2\pi j}{n}} \Rightarrow g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{i(k+n)\,s}$$

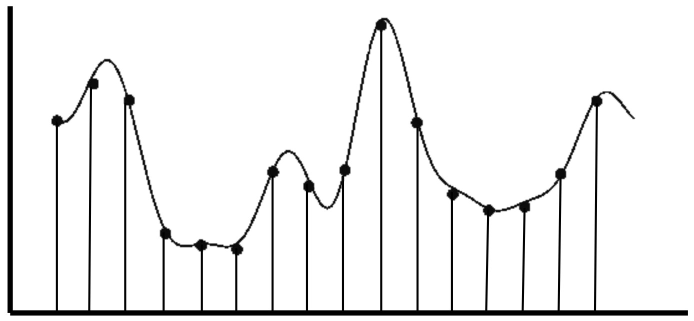
we would also get a continuous function that interpolates the n discrete samples.

Up-Sampling

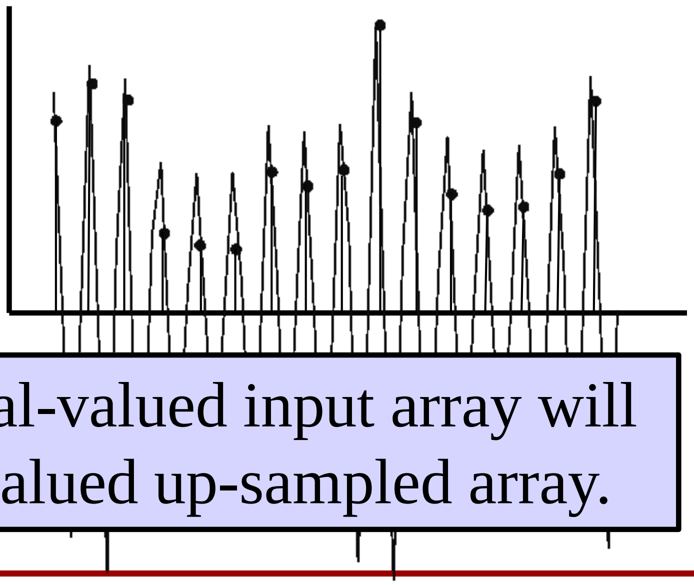
Word of Warning:

The difference is in how the array is interpolated:

$$g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{iks}$$



$$g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{i(k+n)\,s}$$



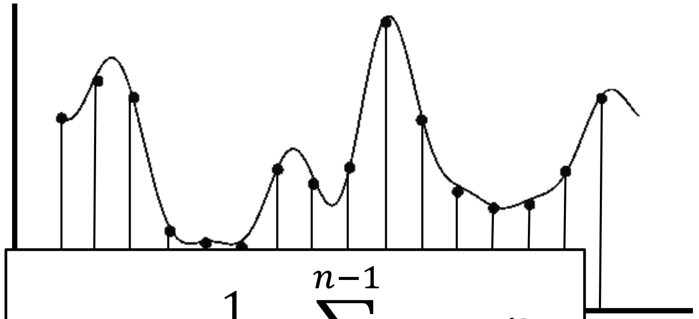
In fact, if not done carefully, a real-valued input array will not necessarily generate a real-valued up-sampled array.

Up-Sampling

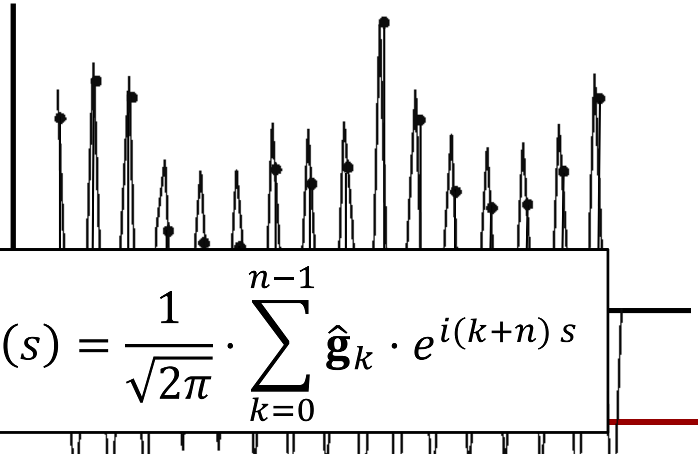
Word of Warning:

Extrapolating the discrete samples is an under-constrained problem, and so there are many different solutions.

In practice, we would like the “smoothest” possible fit, so we would like to minimize the contribution of high frequency terms



$$g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{iks}$$



$$g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot e^{i(k+n)\,s}$$

Up-Sampling

Word of Warning:

Thus, the "best" fit is obtained using the function:

$$g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=-n/2+1}^{n/2} \hat{\mathbf{g}}_k \cdot e^{iks}$$

A simple algorithm for implementing this is:

1. Compute the n Fourier coefficients of $\mathbf{g}$
2. Generate an array of $2n$ Fourier coefficients:

$$\hat{\mathbf{h}}_k = \begin{cases} \hat{\mathbf{g}}_k & -n/2 \le k \le n/2 \\ 0 & \text{otherwise} \end{cases}$$

3. Compute the inverse Fourier transform to get back the $2n$-dimensional array $\mathbf{h}$

Differentiation

Given an n-dimensional array $\mathbf{g}$, how do we compute the derivative of $\mathbf{g}$?

Finite Differences:

Define the derivative at some index j as the average of the discrete left and right derivatives:

$$\mathbf{g}'_j = \frac{(\mathbf{g}_{j+1} - \mathbf{g}_j) + (\mathbf{g}_j - \mathbf{g}_{j-1})}{2} = \frac{\mathbf{g}_{j+1} - \mathbf{g}_{j-1}}{2}$$

Differentiation

Given an n-dimensional array $\mathbf{g}$, how do we compute the derivative of $\mathbf{g}$?

Finite Differences:

Define the derivative at some index j as the average of the discrete left and right derivatives:

$$\mathbf{g}'_j = \frac{\mathbf{g}_{j+1} - \mathbf{g}_{j-1}}{2}$$

Continuous Differentiation:

Fit a continuous function to the samples and take the derivative of the continuous function.

Differentiation

Continuous Differentiation:

Fit a continuous function to the samples and take the derivative of the continuous function.

Fit a continuous function to $\mathbf{g}$ by setting:

$$g(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=-n/2+1}^{n/2} \hat{\mathbf{g}}_k \cdot e^{iks}$$

Since $\frac{\partial}{\partial s} e^{\lambda s} = \lambda e^{\lambda s}$ we get:

$$g'(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=-n/2+1}^{n/2} \hat{\mathbf{g}}_k \cdot ik \cdot e^{iks}$$

Differentiation

Continuous Differentiation:

$$g'(s) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=-n/2+1}^{n/2} \hat{\mathbf{g}}_k \cdot ik \cdot e^{iks}$$

⇒ Get the values of the continuous derivative by multiplying the k-th Fourier coefficient of $\mathbf{g}$ by ik:

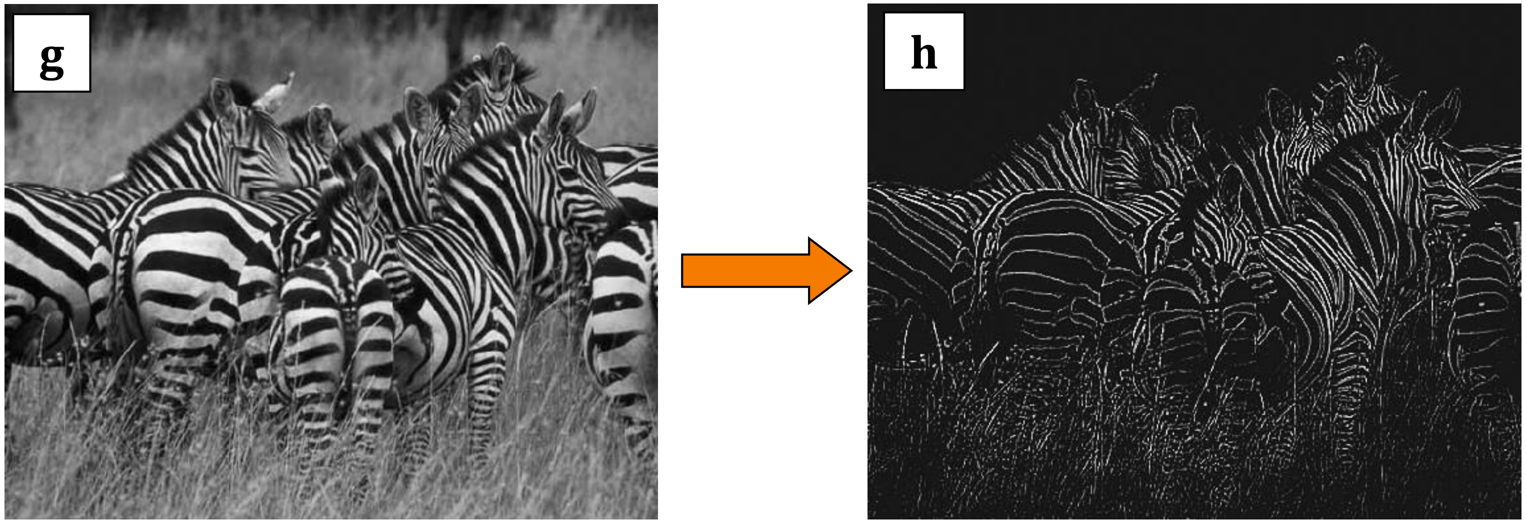
$$\hat{\mathbf{g}}'_k = ik \cdot \hat{\mathbf{g}}_k$$

Note:
As with up-sampling, there are many continuous functions we could fit to the discrete samples.
We use the smoothest one.

Boundary Detection

To compute the boundary of a grid $\mathbf{g}$, we would like to measure how much the grid $\mathbf{g}$ is changing at every index.

Specifically, we would like to define a grid $\mathbf{h}$ such that $\mathbf{h}_{jk}$ measures the "extent of change" of $\mathbf{g}$ at the index (j, k).



Boundary Detection

Gradient Method:

One way to measure the extent change is by computing the gradient of **g** at every point and setting:

$$\mathbf{h}_{jk} = \left\|\nabla \mathbf{g}_{jk}\right\|$$

To compute the gradient, we need to compute the partial derivatives of **g** at every point.

Boundary Detection

Gradient Method:

This can be done with the FFT:

1. Compute the Fourier coefficients of $\mathbf{g}$
2. Generate the two grids corresponding to the Fourier coefficients of the partial derivatives:
$$\hat{\mathbf{g}}^x_{jk} = \hat{\mathbf{g}}_{jk} \cdot ij$$
$$\hat{\mathbf{g}}^y_{jk} = \hat{\mathbf{g}}_{jk} \cdot ik$$
3. Compute the inverse Fourier transforms to get the grids of partial derivatives $\mathbf{g}^x$ and $\mathbf{g}^y$.
4. Set $\mathbf{h}$ to be the grid of gradient lengths:
$$\mathbf{h}_{jk} = \sqrt{\left(\mathbf{g}^x_{jk}\right)^2 + \left(\mathbf{g}^y_{jk}\right)^2}$$

Boundary Detection

Laplacian Method:

An alternate way to measure the rate of change is to compute the difference between the original grid and a smoothed version of the grid.

Boundary Detection

Laplacian Method:

An alternate way to measure the rate of change is to compute the difference between the original grid and a smoothed version of the grid.

A measure of this difference can be obtained by computing the Laplacian (the sum of unmixed) partial derivatives:

$$\Delta f = f_{xx} + f_{yy}$$

Boundary Detection

Laplacian Method:

This can be done with the FFT:

1. Compute the Fourier coefficients of $\mathbf{g}$
2. Generate the grid corresponding to the Fourier coefficients of the Laplacian:

$$\hat{\mathbf{h}}_{jk} = \left(\hat{\mathbf{g}}^{xx} + \hat{\mathbf{g}}^{yy}\right)_{jk} = \hat{\mathbf{g}}_{jk} \cdot \left((ij)^2 + (ik)^2 \right)$$
$$= -\hat{\mathbf{g}}_{jk} \cdot \left(j^2 + k^2\right)$$

3. Set $\mathbf{h}$ to be the inverse Fourier transform of the Laplacian Fourier coefficients.

Gaussian Sharpening

How do we undo the effects of Gaussian-smoothing a grid?

We perform Gaussian smoothing of $\mathbf{g}$ by:

1. Computing the Fourier transforms of $\mathbf{g}$ and the Gaussian filter $\mathbf{f}$
2. Multiplying the Fourier coefficients of $\mathbf{g}$ by the Fourier coefficients of $\mathbf{f}$
3. Computing the inverse Fourier transform

Gaussian Sharpening

How do we undo the effects of Gaussian-smoothing a grid?

We un-perform Gaussian smoothing of **g** by:

1. Computing the Fourier transforms of **g** and the Gaussian grid **f**
2. Multiplying the Fourier coefficients of **g** by the <u>reciprocals</u> of the Fourier coefficients of **f**
3. Computing the inverse Fourier transform

This is doable as long as the Fourier coefficients of **f** are non-zero.

This is numerically stable as long as the Fourier coefficients of **f** are not too small.

Outline

The FFT Algorithm

Multi-Dimensional FFTs

More Applications

Real FFTs

Real FFTs

So far, we have considered the Fourier transform of complex valued functions. What happens when the values of the function are all real?

$$g(\theta) - \overline{g(\theta)} = 0$$

Real FFTs

If we write out the function g in terms of its Fourier decomposition, we get:

$$g(\theta) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{k=-\infty}^{\infty} \hat{\mathbf{g}}_k \cdot e^{ik\theta}$$

Using the fact that $g(\theta) - \bar{g}(\theta) = 0$ we get:

$$0 = \frac{1}{\sqrt{2\pi}} \cdot \left(\sum_{k=-\infty}^{\infty} \hat{\mathbf{g}}_k \cdot e^{ik\theta} - \sum_{k=-\infty}^{\infty} \bar{\hat{\mathbf{g}}}_k \cdot e^{-ik\theta} \right)$$

This can be re-written as:

$$0 = \frac{1}{\sqrt{2\pi}} \cdot \left(\sum_{k=-\infty}^{\infty} \hat{\mathbf{g}}_k \cdot e^{ik\theta} - \sum_{k=-\infty}^{\infty} \bar{\hat{\mathbf{g}}}_{-k} \cdot e^{ik\theta} \right)$$

Real FFTs

$$0 = \frac{1}{\sqrt{2\pi}} \cdot \left(\sum_{k=-\infty}^{\infty} \hat{\mathbf{g}}_k \cdot e^{ik\theta} - \sum_{k=-\infty}^{\infty} \bar{\hat{\mathbf{g}}}_{-k} \cdot e^{ik\theta} \right)$$

Simplifying this equation we get:

$$0 = \sum_{k=-\infty}^{\infty} \left(\hat{\mathbf{g}}_k - \bar{\hat{\mathbf{g}}}_{-k} \right) \cdot e^{ik\theta}$$

But since complex exponentials are linearly independent, this implies that all the Fourier coefficients are equal to zero:

$$0 = \hat{\mathbf{g}}_k - \bar{\hat{\mathbf{g}}}_{-k}$$

$$\Updownarrow$$

$$\hat{\mathbf{g}}_k = \bar{\hat{\mathbf{g}}}_{-k}$$

Real FFTs

$$\hat{\mathbf{g}}_k = \overline{\hat{\mathbf{g}}}_{-k}$$

⇒ When the function g is real-values, the k-th and $(-k)$-th Fourier coefficient are conjugate.

Real FFTs

Although this discussion is true for real functions, a similar argument shows that for real-valued arrays $\mathbf{g} \in \mathbb{R}^n$, the Fourier coefficients have the property that:

$$\hat{\mathbf{g}}_k = \overline{\hat{\mathbf{g}}}_{n-k}$$

Real FFTs

For real-valued arrays:

- In 1D we have:

$$\hat{\mathbf{g}}_j = \overline{\hat{\mathbf{g}}}_{n-j}$$

- In 2D we have:

$$\hat{\mathbf{g}}_{jk} = \overline{\hat{\mathbf{g}}}_{n-j,n-k}$$

- In 3D we have:

$$\hat{\mathbf{g}}_{jkl} = \overline{\hat{\mathbf{g}}}_{n-j,n-k,n-l}$$

⇒ When the array is real-valued, we only have to compute and store half of the coefficients.

Real FFTs

Recall:

The Fourier basis $\{\mathbf{z}^0, \dots, \mathbf{z}^{n-1}\}$ satisfies:

$$\mathbf{z}_j^{-k} = \mathbf{z}_{-j}^{k} = \bar{\mathbf{z}}_j^{k}$$

Real FFTs

Recall:

In the Fourier domain, we correlate a signal $\mathbf{g}$ with a signal $\mathbf{f}$ by multiplying the Fourier coefficients of g by the conjugate of the Fourier coefficients of $\mathbf{f}$:

$$\mathbf{f} = \sum_{k=0}^{n-1} \hat{\mathbf{f}}_k \cdot \mathbf{z}^k \qquad\qquad \mathbf{g} = \sum_{k=0}^{n-1} \hat{\mathbf{g}}_k \cdot \mathbf{z}^k$$

$$\Downarrow$$

$$(f \star g)_\alpha = \sum_{j=1}^{n} \hat{\mathbf{f}}_j \cdot \overline{\hat{\mathbf{g}}}_j \cdot \mathbf{x}_\alpha^j$$

Real FFTs

Recall:

To compute the convolution of a filter $\mathbf{f}$, with a signal $\mathbf{g}$ we reflect $\mathbf{f}$ through the origin and then correlate:

$$\mathbf{g} * \mathbf{f} = \mathbf{g} \star \tilde{\mathbf{f}} \qquad \text{w/} \;\; \tilde{\mathbf{f}}_k = \mathbf{f}_{-k}$$

Real FFTs

$$\mathbf{g} * \mathbf{f} = \mathbf{g} \star \tilde{\mathbf{f}} \qquad \text{w/} \quad \tilde{\mathbf{f}}_k = \mathbf{f}_{-k}$$

Expressing $\mathbf{f}$ in terms of the Fourier basis gives:

$$\mathbf{f}_j = \sum_{k=0}^{n-1} \hat{\mathbf{f}}_k \cdot \mathbf{z}_j^k$$

Reflecting through the origin gives:

$$\begin{aligned}\mathbf{f}_{-j} &= \sum_{k=0}^{n-1} \hat{\mathbf{f}}_k \cdot \mathbf{z}_{-j}^k = \sum_{k=0}^{n-1} \hat{\mathbf{f}}_k \cdot \bar{\mathbf{z}}_j^k \\ &= \sum_{k=0}^{n-1} \hat{\mathbf{f}}_{-k} \cdot \bar{\mathbf{z}}_j^{-k} \\ &= \sum_{k=0}^{n-1} \bar{\hat{\mathbf{f}}}_k \cdot \mathbf{z}_j^k\end{aligned}$$

Real FFTs

$$\mathbf{g} * \mathbf{f} = \mathbf{g} \star \tilde{\mathbf{f}} \qquad \text{w/} \;\; \tilde{\mathbf{f}}_k = \mathbf{f}_{n-k}$$

Expressing **f** in terms of the Fourier basis gives:

$$\mathbf{f}_j = \sum_{k=0}^{n-1} \hat{\mathbf{f}}_k \cdot \mathbf{z}_j^k$$

Reflecting through the origin gives:

If f is real-valued, reflecting f through the origin is equivalent to conjugating the Fourier coefficients of f.

In the Fourier domain, convolution is multiplication (w/o conjugation).

$$= \sum_{k=0}^{n-1} \overline{\hat{\mathbf{f}}}_k \cdot \mathbf{z}_j^k$$