FFTs in Graphics and Vision

Moving Dot Products

Outline

Review

Moving Dot Products:

- One-Dimensional (Continuous)
- One-Dimensional (Discrete)
- Higher-Dimensional
- Computational Complexity

Representations

A <u>representation</u> of a group G on a vector space V, denoted (ρ, V), is a map ρ that sends every element in G to an invertible linear transformation on V, satisfying:

$$\rho(g \cdot h) = \rho(g) \cdot \rho(h) \quad \forall g, h \in G.$$

Sub-Representation

Given a representation (ρ, V) of a group G, if there exists a subspace $W \subset V$ such that the representation fixes W:

$$\rho_g(w) \in W \qquad \forall g \in G;\ w \in W$$

then we say that W is a <u>sub-representation</u> of V.

Irreducible Representations

Given a representation (ρ, V) of a group G, the representation is said to be <u>irreducible</u> if the only subspaces of V that are sub-representations are:

$$W = V \quad \text{and} \quad W = \emptyset$$

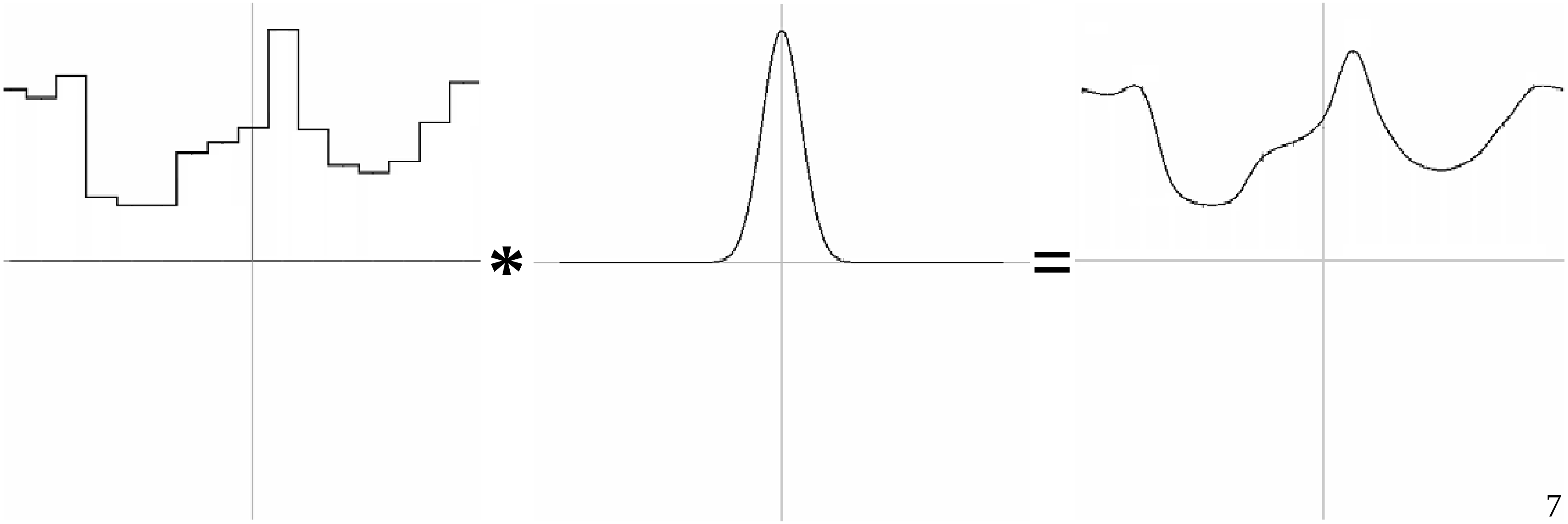
Schur's Lemma (Corollary)

If (ρ, V) is an irreducible, (unitary), representation of a commutative group G, then V must be one-dimensional.

Why do we care?

In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

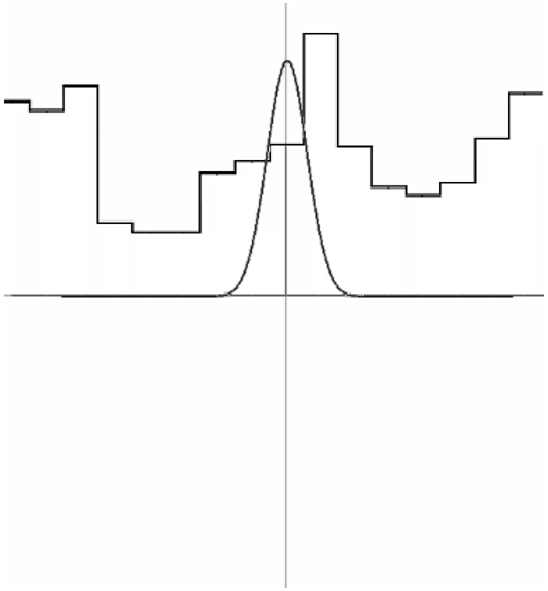
E.g. Smoothing:

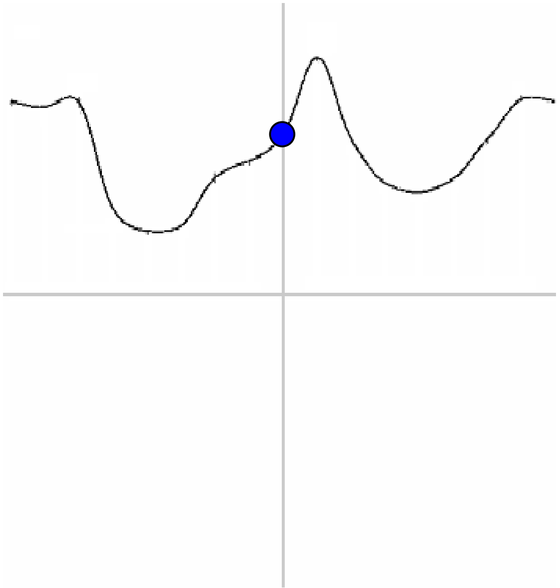


Why do we care?

In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

E.g. Smoothing:

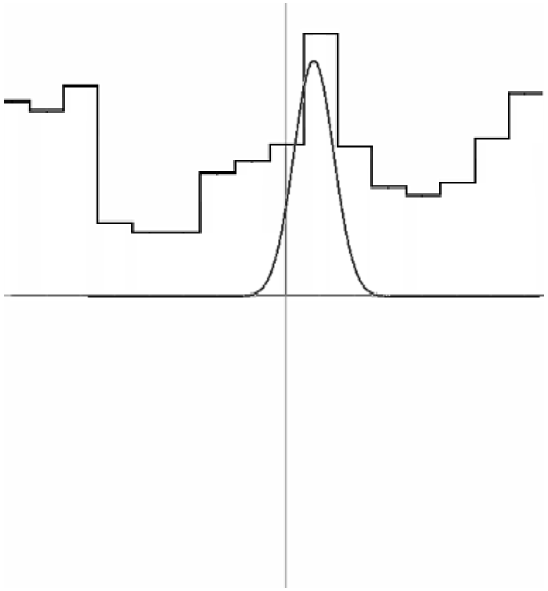


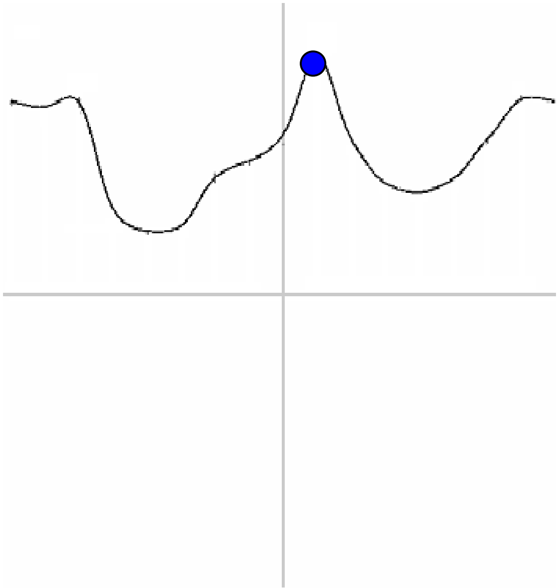


Why do we care?

In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

E.g. Smoothing:

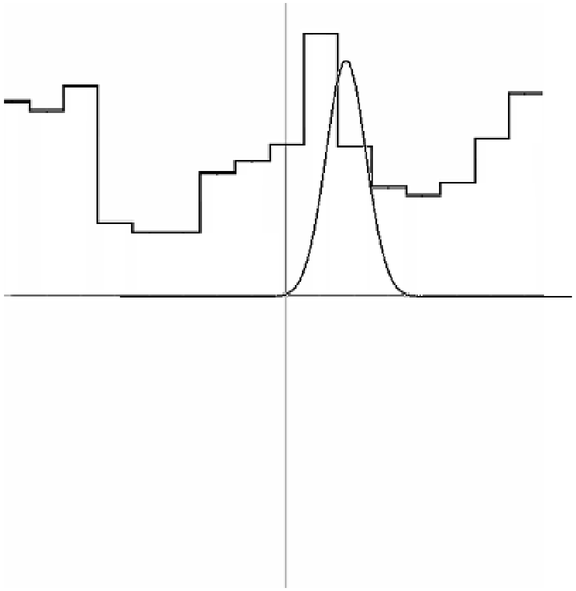


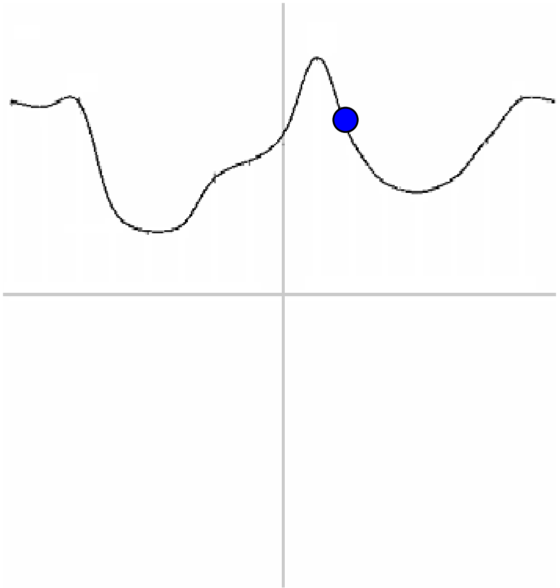


Why do we care?

In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

E.g. Smoothing:

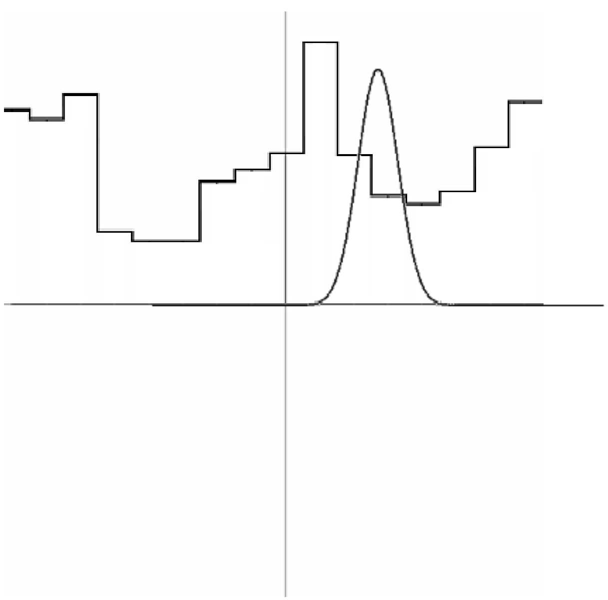


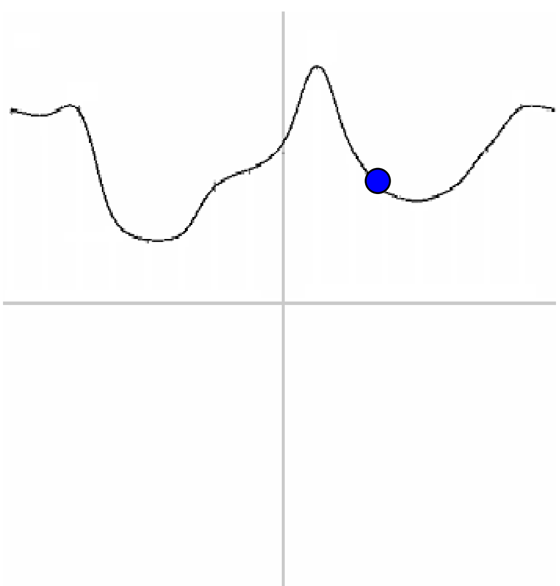


Why do we care?

In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

E.g. Smoothing:

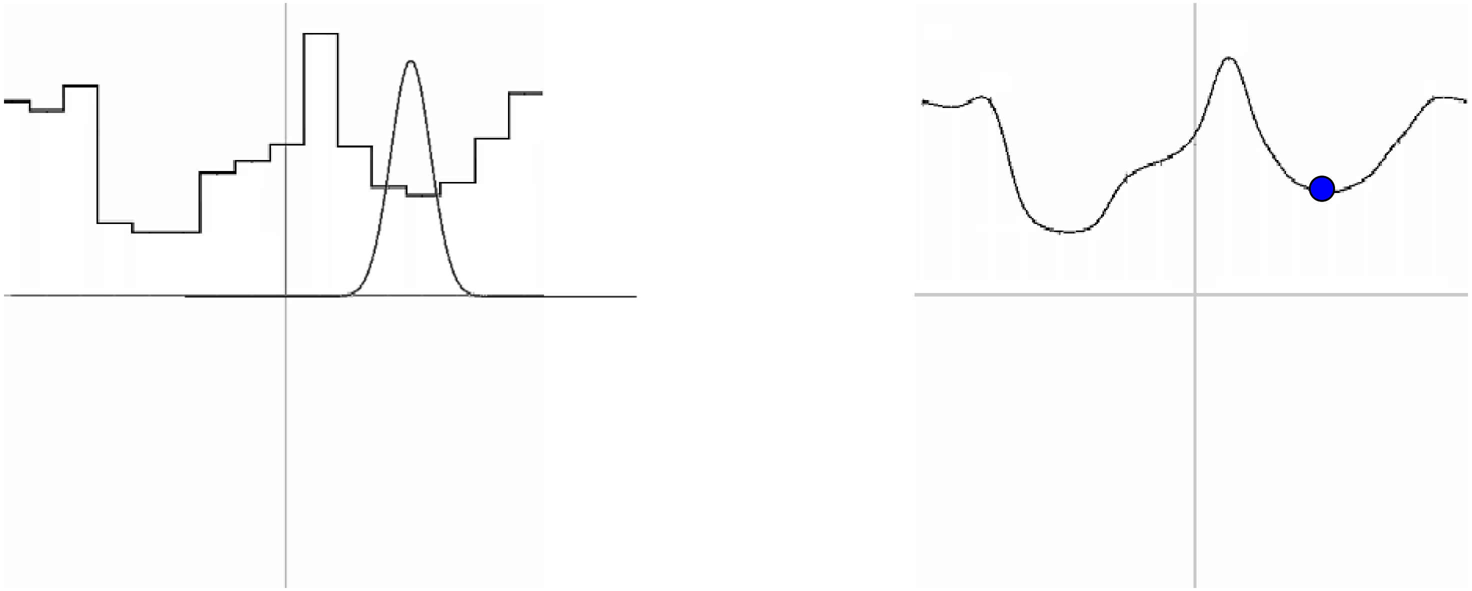




Why do we care?

In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

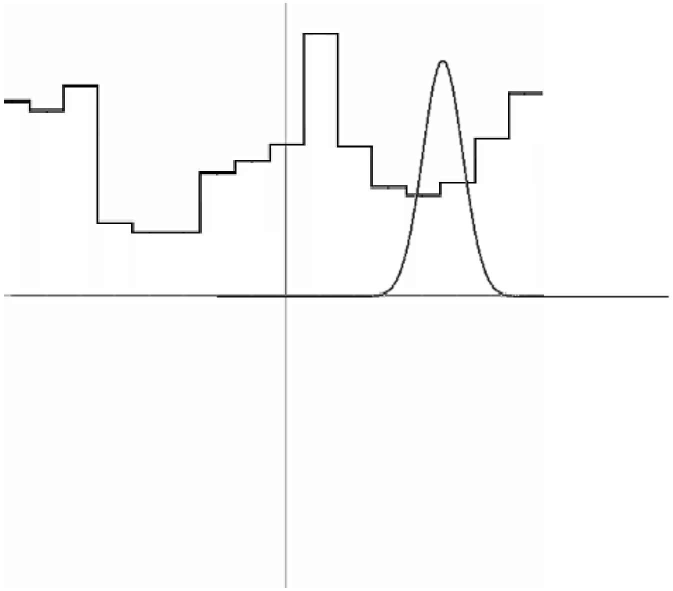
E.g. Smoothing:

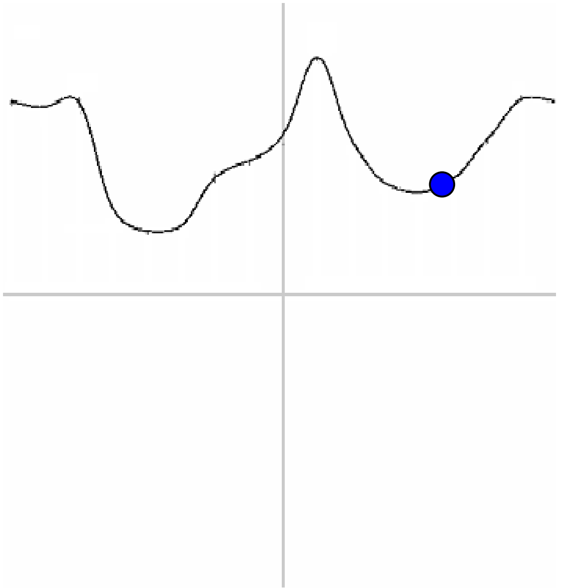


Why do we care?

In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

E.g. Smoothing:

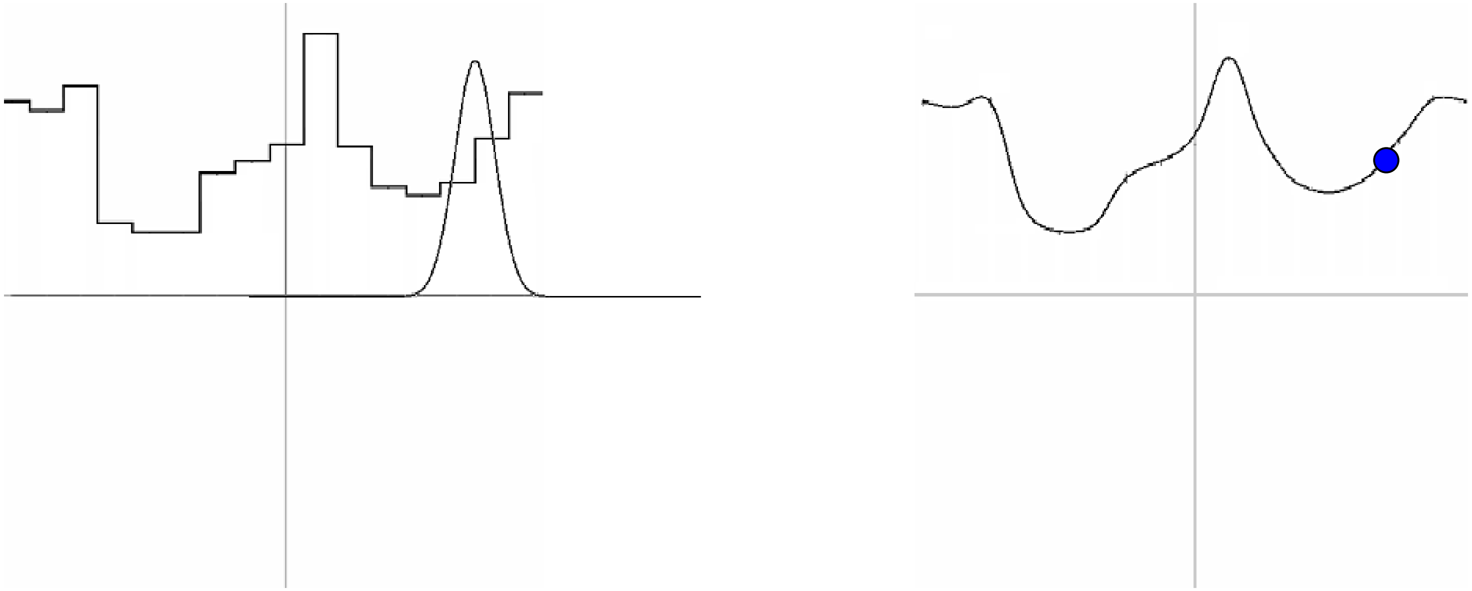




Why do we care?

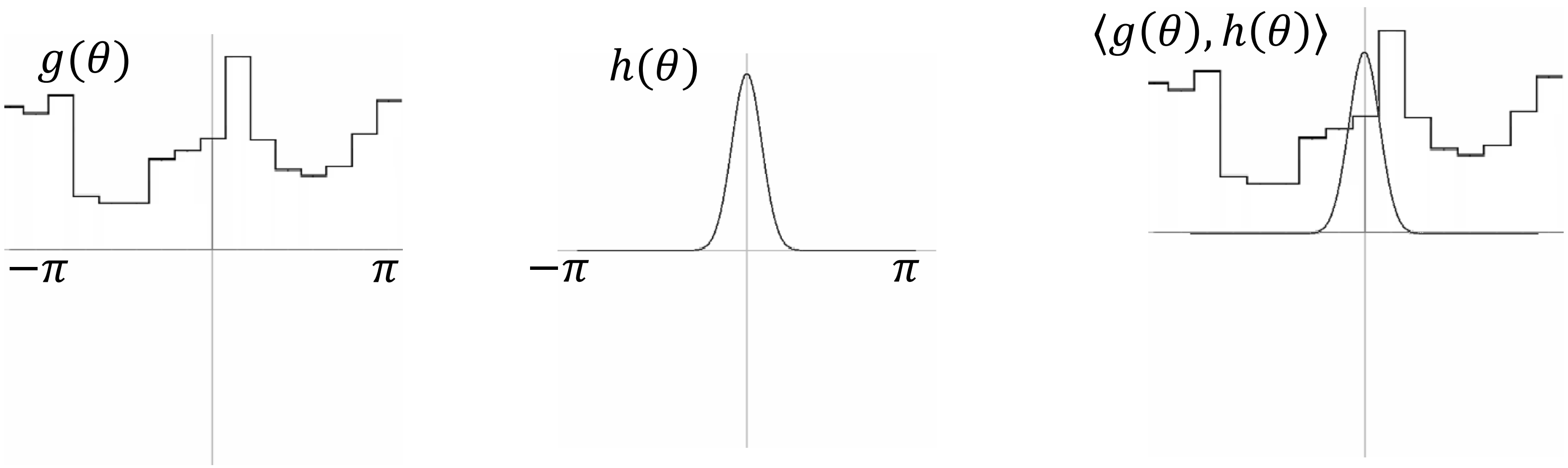
In signal/image/voxel processing, we are often interested in applying a filter to some initial data.

E.g. Smoothing:



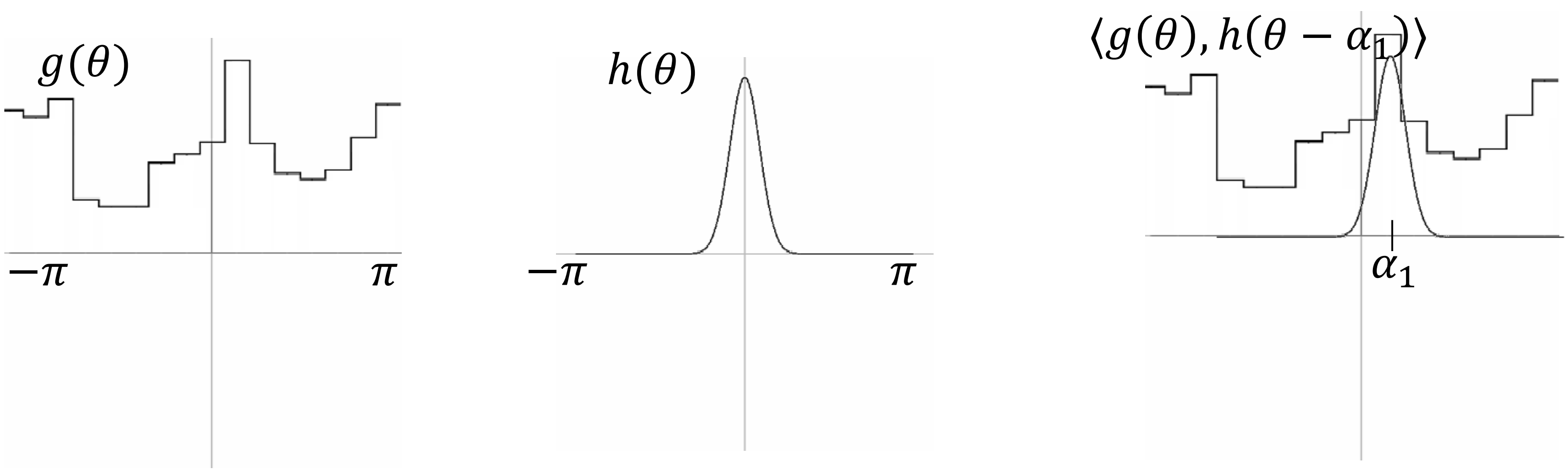
Smoothing

What we are really doing is computing a moving inner product:



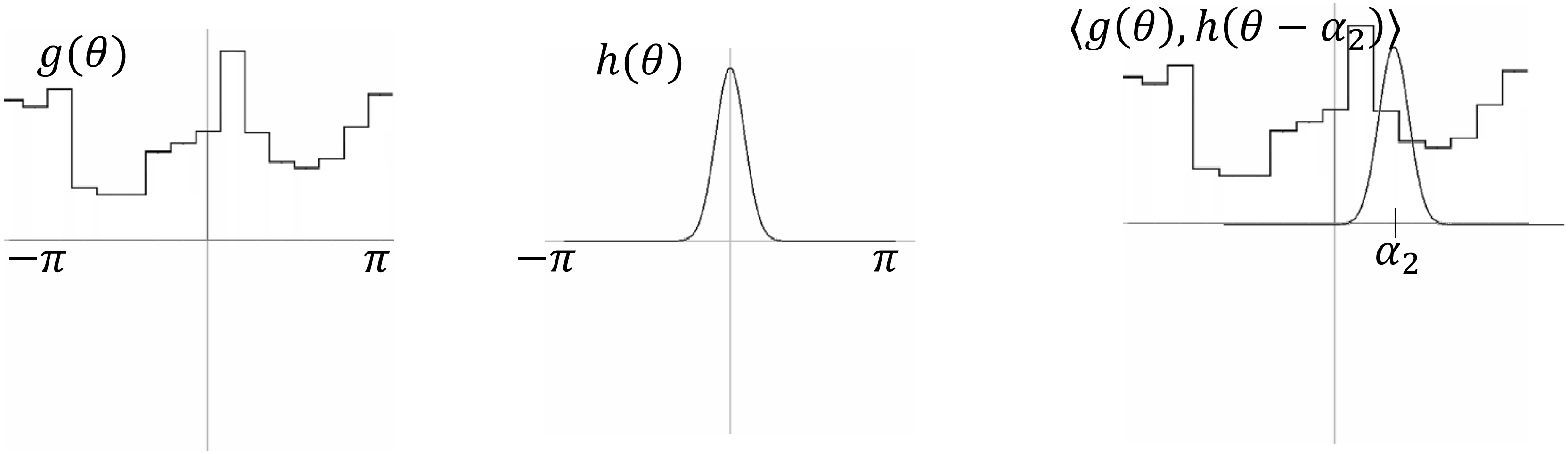
Smoothing

What we are really doing is computing a moving inner product:



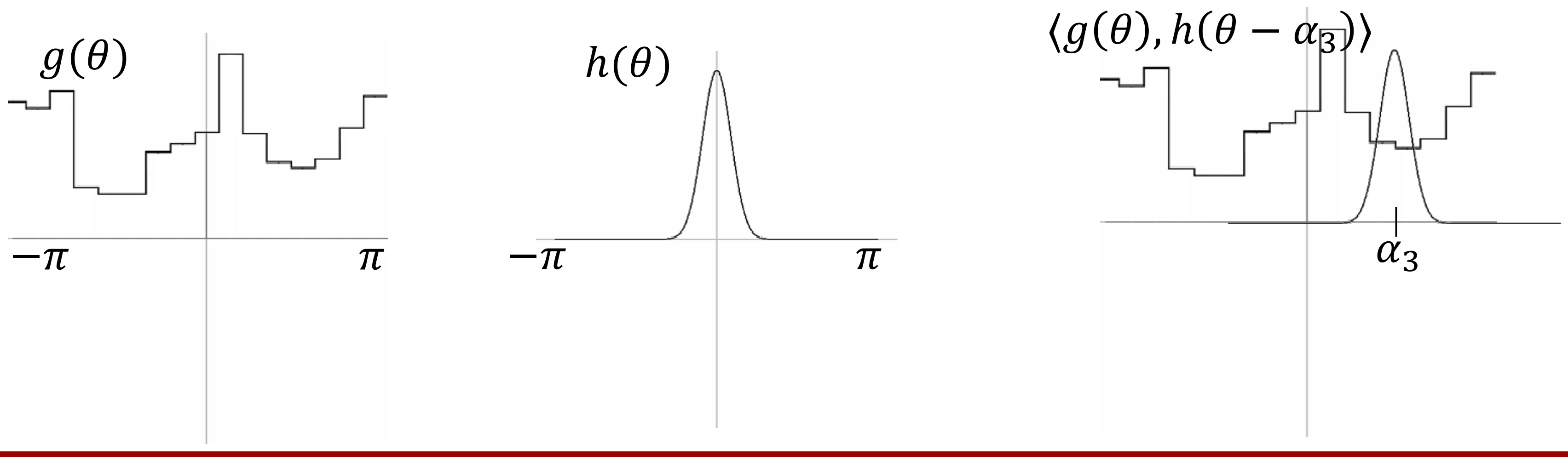
Smoothing

What we are really doing is computing a moving inner product:



Smoothing

What we are really doing is computing a moving inner product:

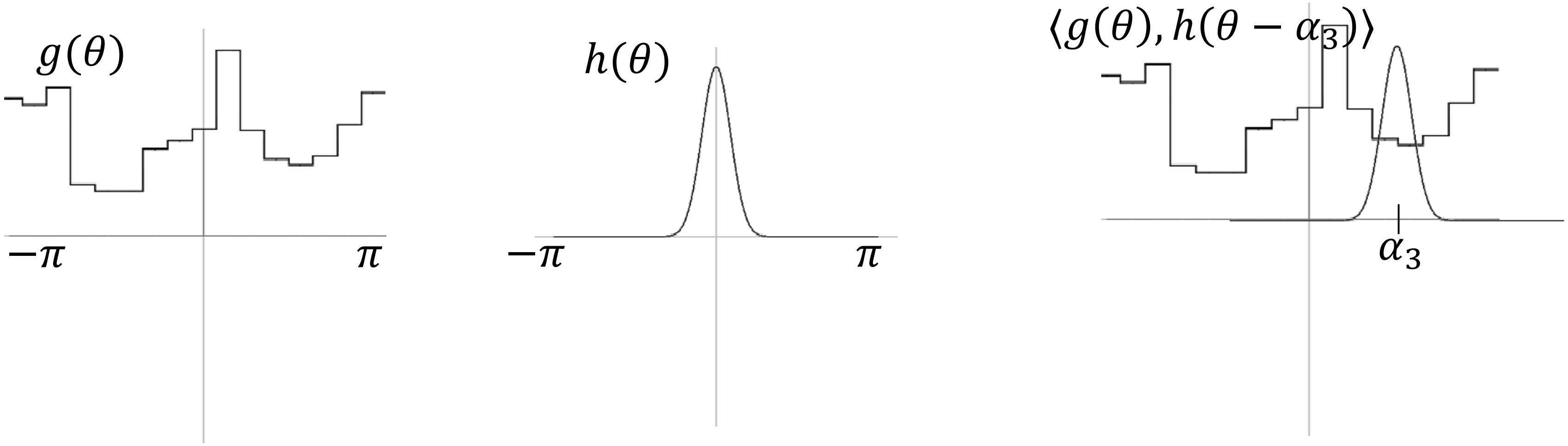


Smoothing

We can write out the operation of smoothing a signal g by a filter h as:

$$(g \star h)(\alpha) = \langle g, \rho_\alpha(h) \rangle$$

where ρ_α is the linear transformation that translates a periodic function by α.



Moving Dot Products

We can think of this as a representation:

- V is the space of periodic functions on the line
- G is the group of real numbers in $[0,2\pi)$
- ρ_α is the representation translating a function by α.

Moving Dot Products

We can think of this as a representation:

- V is the space of periodic functions on the line
- G is the group of real numbers in $[0,2\pi)$
- ρ_α is the representation translating a function by α.

This is a representation of a commutative group…

Smoothing

$\Rightarrow$ There exist orthogonal one-dimensional (complex) subspaces $V_1, \cdots, V_n \subset V$ that are the irreducible representations of V.

Setting $\phi_i \in V_i$ to be a unit-vector, we know that the group acts on ϕ_i by scalar multiplication:

$$\rho_\alpha(\phi_i) = \lambda_i(\alpha) \cdot \phi_i$$

Note:

Since the V_i are orthogonal, the basis $\{\phi_1, \cdots \phi_n\}$ is orthonormal.

Smoothing

Setting $\phi_i \in V_i$ to be a unit-vector, we know that the group acts on ϕ_i by scalar multiplication:

$$\rho_\alpha(\phi_i) = \lambda_i(\alpha) \cdot \phi_i$$

We can write out the functions $g, h \in V$ as:

$$g(\theta) = \hat{g}_1 \cdot \phi_1(\theta) + \cdots + \hat{g}_n \cdot \phi_n(\theta)$$
$$h(\theta) = \hat{h}_1 \cdot \phi_1(\theta) + \cdots + \hat{h}_n \cdot \phi_n(\theta)$$

with $\hat{g}_i, \hat{h}_i \in \mathbb{C}$.

Smoothing

Then the moving dot-product can be written as:

$$(g \star h)(\alpha) = \langle g, \rho_\alpha(h) \rangle$$

Smoothing

$$(g \star h)(\alpha) = \langle g, \rho_\alpha(h) \rangle$$

Expanding in the basis $\{\phi_1, \cdots, \phi_n\}$:

$$(g \star h)(\alpha) = \left\langle \sum_{j=1}^{n} \hat{g}_j \phi_j \, , \rho_\alpha \left(\sum_{k=1}^{n} \hat{h}_k \phi_k \right) \right\rangle$$

Smoothing

$$(g \star h)(\alpha) = \left\langle \sum_{j=1}^{n} \hat{g}_j \phi_j \, , \rho_\alpha \left(\sum_{k=1}^{n} \hat{h}_k \phi_k \right) \right\rangle$$

By linearity of ρ_α:

$$(g \star h)(\alpha) = \left\langle \sum_{j=1}^{n} \hat{g}_j \phi_j \, , \sum_{k=1}^{n} \hat{h}_k \rho_\alpha(\phi_k) \right\rangle$$

Smoothing

$$(g \star h)(\alpha) = \left\langle \sum_{j=1}^{n} \hat{g}_j \phi_j \, , \sum_{k=1}^{n} \hat{h}_k \rho_\alpha(\phi_k) \right\rangle$$

By linearity of the inner product in the first term:

$$(g \star h)(\alpha) = \sum_{j=1}^{n} \hat{g}_j \left\langle \phi_j, \sum_{k=1}^{n} \hat{h}_k \rho_\alpha(\phi_k) \right\rangle$$

Smoothing

$$(g \star h)(\alpha) = \sum_{j=1}^{n} \hat{g}_j \left\langle \phi_j, \sum_{k=1}^{n} \hat{h}_k \rho_\alpha(\phi_k) \right\rangle$$

By conjugate-linearity in the second term:

$$(g \star h)(\alpha) = \sum_{j,k=1}^{n} \hat{g}_j \overline{\hat{h}}_k \langle \phi_j, \rho_\alpha(\phi_k) \rangle$$

Smoothing

$$(g \star h)(\alpha) = \sum_{j,k=1}^{n} \hat{g}_j \bar{\hat{h}}_k \langle \phi_j, \rho_\alpha(\phi_k) \rangle$$

Because ρ_α is scalar multiplication in V_i:

$$(g \star h)(\alpha) = \sum_{j,k=1}^{n} \hat{g}_j \bar{\hat{h}}_k \langle \phi_j, \lambda_k(\alpha) \phi_k \rangle$$

Smoothing

$$(g \star h)(\alpha) = \sum_{j,k=1}^{n} \hat{g}_j \bar{\hat{h}}_k \langle \phi_j, \lambda_k(\alpha)\phi_k \rangle$$

Again, by conjugate-linearity in the second term:

$$(g \star h)(\alpha) = \sum_{j,k=1}^{n} \hat{g}_j \bar{\hat{h}}_k \overline{\lambda_k(\alpha)} \langle \phi_j, \phi_k \rangle$$

Smoothing

$$(g \star h)(\alpha) = \sum_{j,k=1}^{n} \hat{g}_j \bar{\hat{h}}_k \overline{\lambda_k(\alpha)} \langle \phi_j, \phi_k \rangle$$

And finally, by the orthonormality of $\{\phi_1, \cdots, \phi_n\}$:

$$(g \star h)(\alpha) = \sum_{j=1}^{n} \hat{g}_j \bar{\hat{h}}_j \overline{\lambda_j(\alpha)}$$

Smoothing

$$(g \star h)(\alpha) = \sum_{j=1}^{n} \hat{g}_j \bar{\hat{h}}_j \overline{\lambda_j(\alpha)}$$

This implies that we can compute the moving dot-product by multiplying the coefficients of g and h.

Convolution/Correlation in the spatial domain is multiplication in the frequency domain!

Smoothing

What is $\lambda_j(\alpha)$?

Smoothing

What is $\lambda_j(\alpha)$?

Since the representation is unitary, $\left|\lambda_j(\alpha)\right| = 1$.

$$\Downarrow$$

$$\exists \tilde{\lambda}_j : [0,2\pi) \to \mathbb{R} \quad \text{s.t.} \quad \lambda_j(\alpha) = e^{i\tilde{\lambda}_j(\alpha)}$$

Smoothing

What is $\lambda_j(\alpha)$?

$$\lambda_j(\alpha) = e^{i\tilde{\lambda}_j(\alpha)} \text{ for some } \tilde{\lambda}_j : [0,2\pi) \to \mathbb{R}.$$

Since it's a representation:

$$\Downarrow$$

$$\lambda_j(\alpha+\beta) = \lambda_j(\alpha) \cdot \lambda_j(\beta) \quad \forall \alpha, \beta \in [0,2\pi)$$

$$\Downarrow$$

$$\tilde{\lambda}_j(\alpha+\beta) = \tilde{\lambda}_j(\alpha) + \tilde{\lambda}_j(\beta)$$

$$\Downarrow$$

$$\exists \kappa_j \in \mathbb{R} \quad s.t. \quad \tilde{\lambda}_j(\alpha) = \kappa_j \cdot \alpha$$

Smoothing

<u>What is $\lambda_j(\alpha)$</u>?

$$\lambda_j(\alpha) = e^{i\kappa_j\alpha} \text{ for some } \kappa_j \in \mathbb{R}.$$

Since it's a representation:

$$\Downarrow$$

$$1 = \lambda_j(2\pi) = e^{i\kappa_j 2\pi}$$

$$\Downarrow$$

$$\kappa_j \in \mathbb{Z}$$

Smoothing

Thus, the correlation of the signals $g, h: S^1 \to \mathbb{C}$ can be expressed as:

$$(g \star h)(\alpha) = \sum_{j=1}^{n} \hat{g}_j \bar{\hat{h}}_j e^{-i\kappa_j \alpha}$$

where $\kappa_j \in \mathbb{Z}$.

Outline

Review

Moving Dot Products:

- One-Dimensional (Continuous)
- One-Dimensional (Discrete)
- Higher-Dimensional
- Computational Complexity

Moving Dot Products (Periodic Functions)

Let's consider the case of periodic functions in more detail:

Moving Dot Products (Periodic Functions)

Let's consider the case of periodic functions in more detail:

- V is the space of periodic functions on the line
- G is the group of real numbers in $[0,2\pi)$
- ρ_α is the representation translating a function by α:

$$\big(\rho_\alpha(f)\big)(\theta) = f(\theta - \alpha)$$

What are the irreducible representations V_k?

What are the corresponding functions $\lambda_k(\alpha)$?

Moving Dot Products (Periodic Functions)

It turns out that the one-dimensional spaces V_k are the spans of the complex exponentials:

$$V_k = \mathrm{Span}\left(e^{ik\theta}\right)$$

Moving Dot Products (Periodic Functions)

It turns out that the one-dimensional spaces V_k are the spans of the complex exponentials:

$$V_k = \mathrm{Span}\left(e^{ik\theta}\right)$$

Given any vector $v \in V_k$, applying ρ_α to v, we get:

$$\begin{aligned}\rho_\alpha(v) &= \rho_\alpha\left(c \cdot e^{ik\theta}\right) \\ &= c \cdot \rho_\alpha\left(e^{ik\theta}\right) \\ &= c \cdot e^{ik(\theta-\alpha)} \\ &= c \cdot e^{ik\theta} \cdot e^{-ik\alpha} \\ &= v \cdot e^{-ik\alpha}\end{aligned}$$

$$\boxed{\lambda_k(\alpha) = e^{-ik\alpha}}$$

Moving Dot Products (Periodic Functions)

Note

The periodic functions:

$$f_k(\theta) = e^{ik\theta}$$

do not have unit norm!

$$\begin{aligned}\|f_k\|^2 &= \int_0^{2\pi} e^{ik\theta} \cdot \overline{e^{ik\theta}}\, d\theta \\ &= \int_0^{2\pi} 1\, d\theta \\ &= 2\pi\end{aligned}$$

Moving Dot Products (Periodic Functions)

Note

The periodic functions:

$$f_k(\theta) = e^{ik\theta}$$

do not have unit norm!

We need to normalize the functions to make them unit-norm:

$$f_k(\theta) = \sqrt{\frac{1}{2\pi}} e^{ik\theta}$$

$$\boxed{\overline{\lambda_k(\alpha)} = e^{ik\alpha} = \sqrt{2\pi} f_k(\theta)}$$

Moving Dot Products (Periodic Functions)

Thus, given two periodic functions on the line, $g(\theta)$ and $h(\theta)$, we can expand:

$$g(\theta) = \sum_{k=-\infty}^{\infty} \hat{g}_k \sqrt{\frac{1}{2\pi}} e^{ik\theta} \quad \text{and} \quad h(\theta) = \sum_{k=-\infty}^{\infty} \hat{h}_k \sqrt{\frac{1}{2\pi}} e^{ik\theta}$$

to get:

$$\begin{aligned}(g \star h)(\alpha) &= \sum_{k=-\infty}^{\infty} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot \overline{\lambda_k}(\alpha) \\ &= \sum_{k=-\infty}^{\infty} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot e^{ik\alpha} \\ &= \sum_{k=-\infty}^{\infty} \sqrt{2\pi} \cdot \hat{g}_k \cdot \overline{\hat{h}_k} \cdot \sqrt{\frac{1}{2\pi}} e^{ik\alpha}\end{aligned}$$

Moving Dot Products (Periodic Functions)

What's really going on here?

If we express a complex number in terms of radius and angle (r, θ), then rotation by α degrees corresponds to the map:

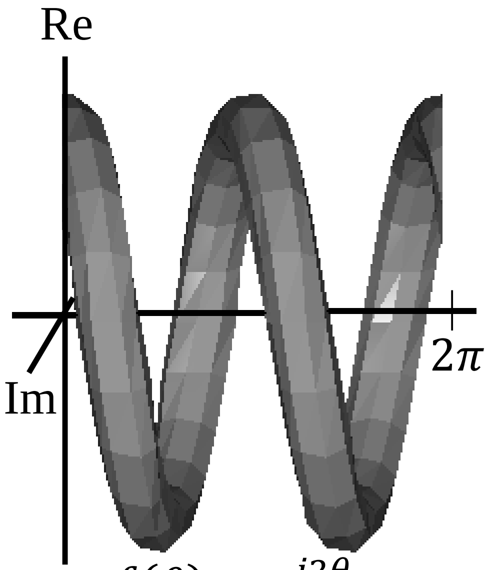
$$(r, \theta) \to (r, \theta + \alpha)$$
$$\Updownarrow$$
$$re^{i\theta} \to re^{i(\theta+\alpha)} = e^{i\alpha} \cdot re^{i\theta}$$

Rotating in the complex plane is the same thing as multiplying by a complex, unit-norm, number.

Moving Dot Products (Periodic Functions)

What's really going on here?

Let's consider the graph of a complex exponential. This is just a helix:



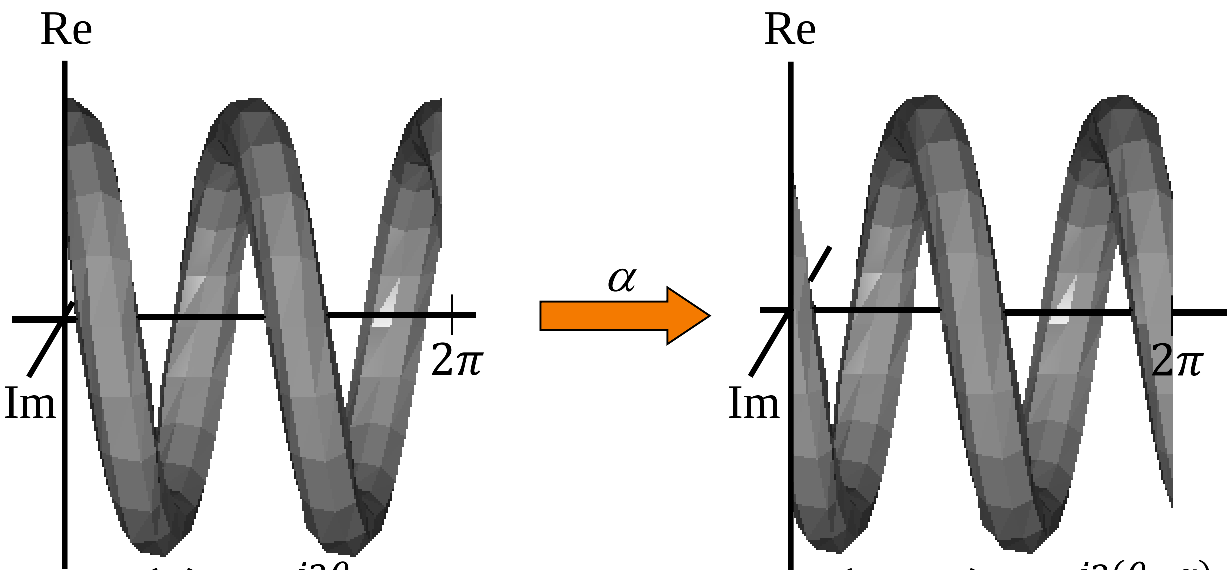
$f(\theta) = e^{i2\theta}$

Moving Dot Products (Periodic Functions)

What’s really going on here?

Let’s consider the graph of a complex exponential. This is just a helix.

If we translate the function by α, we get:



$f(\theta) = e^{i2\theta}$ $\quad$ $f(\theta - \alpha) = e^{i2(\theta-\alpha)}$

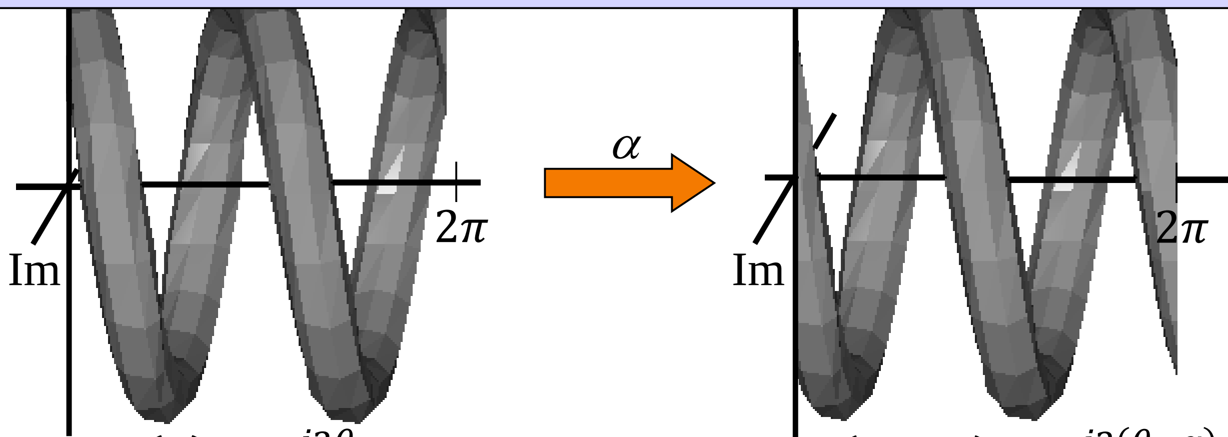
Moving Dot Products (Periodic Functions)

What’s really going on here?

Let’s consider the graph of a complex exponential. This is just a helix.

If we translate the function by α, we get:

Translating a periodic helix along its axis is the same thing as rotating the helix around it.



$f(\theta) = e^{i2\theta}$ $\qquad$ $f(\theta - \alpha) = e^{i2(\theta-\alpha)}$

Outline

Review

Moving Dot Products:

- One-Dimensional (Continuous)
- One-Dimensional (Discrete)
- Higher-Dimensional
- Computational Complexity

Moving Dot Products (Periodic Arrays)

In practice, we don't have infinite precision, and we discretize the function space and the group:

- V is the space of periodic n-dimensional arrays
- G is the group of integers modulo n
- ρ_j is the representation shifting the entries in the array by j positions

What are the irreducible representations V_k?

What are the corresponding functions $\lambda_k(\alpha)$?

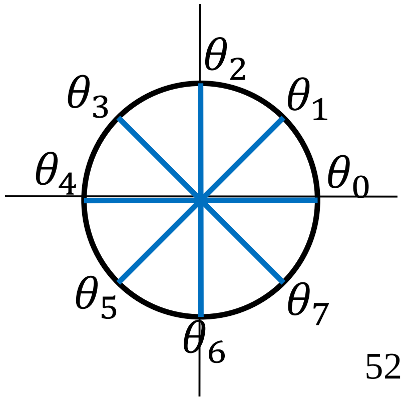
Moving Dot Products (Periodic Arrays)

We set V_k to be the (1D) spaces spanned by the <u>discretizations</u> of the complex exponentials:

$$V_k = \mathrm{Span}(v_k)$$

where v_k is defined by regularly sampling the k-th complex exponential:

$$v_k[\cdot] = \left(e^{ik\theta_0}, \cdots, e^{ik\theta_{n-1}}\right) \quad \text{with } \theta_j = \frac{2\pi j}{n}$$



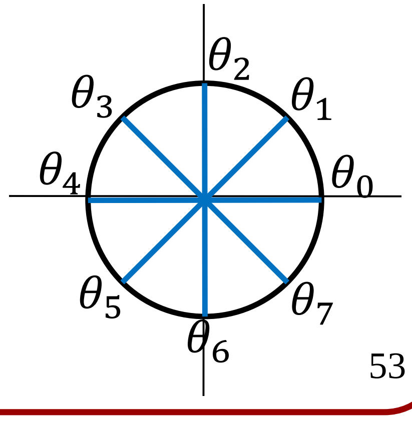
Moving Dot Products (Periodic Arrays)

Applying ρ_α to $v_k[\cdot]$, we get:

$$\rho_\alpha(v_k[\cdot]) = \left(e^{ik\theta_{0-\alpha}}, \cdots, e^{ik\theta_{n-1-\alpha}}\right)$$

We can write out:

$$\begin{aligned}\theta_{j-\alpha} &= \frac{2\pi(j-\alpha)}{n} \\ &= \frac{2\pi j}{n} + \frac{-2\pi\alpha}{n} \\ &= \theta_j + \theta_{-\alpha}\end{aligned}$$



Moving Dot Products (Periodic Arrays)

Applying ρ_α to $v_k[\cdot]$, we get:

$$\rho_\alpha(v_k[\cdot]) = \left(e^{ik\theta_{0-\alpha}}, \cdots, e^{ik\theta_{n-1-\alpha}}\right)$$

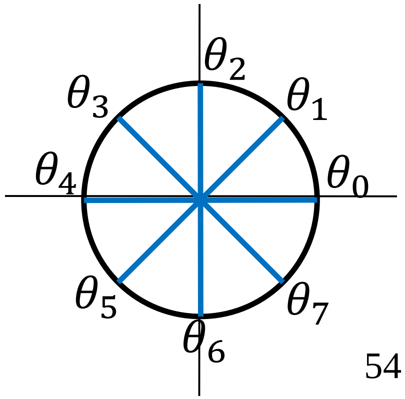
We can write out:

$$\theta_{j-\alpha} = \theta_j + \theta_{-\alpha}$$

So that:

$$\begin{aligned}\rho_\alpha(v_k[\cdot]) &= \left(e^{ik\theta_0} \cdot e^{ik\theta_{-\alpha}}, \cdots, e^{ik\theta_{n-1}} \cdot e^{ik\theta_{-\alpha}}\right) \\ &= e^{ik\theta_{-\alpha}} \cdot v_k[\cdot]\end{aligned}$$

$$\boxed{\overline{\lambda_k}[\alpha] = e^{ik\theta_\alpha}}$$



Moving Dot Products (Periodic Arrays)

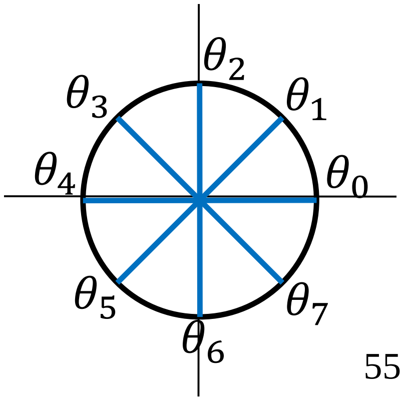
Note 1

The periodic arrays:

$$v_k[\cdot] = \left(e^{ik\theta_0}, \cdots, e^{ik\theta_{n-1}}\right)$$

do not have unit norm!

$$\begin{aligned}\|v_k[\cdot]\|^2 &= \sum_{j=0}^{n-1} v_k[j] \cdot \overline{v_k[j]} \\ &= \sum_{j=0}^{n-1} e^{ik\theta_j} \cdot e^{-ik\theta_j} \\ &= n\end{aligned}$$



Moving Dot Products (Periodic Arrays)

<u>Note 1</u>

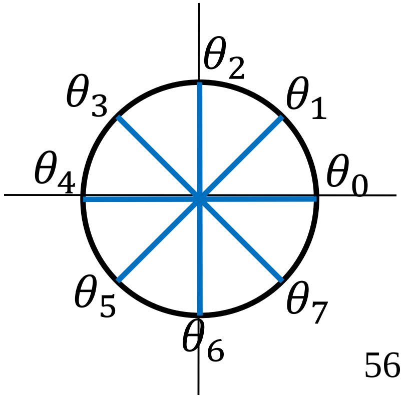
The periodic arrays:

$$v_k[\cdot] = \left(e^{ik\theta_0}, \cdots, e^{ik\theta_{n-1}}\right)$$

do not have unit norm!

We need to normalize these functions to make them unit-norm:

$$v_k[\cdot] = \sqrt{\frac{1}{n}}\left(e^{ik\theta_0}, \cdots, e^{ik\theta_{n-1}}\right)$$



Moving Dot Products (Periodic Arrays)

<u>Note 1</u>

The periodic arrays:

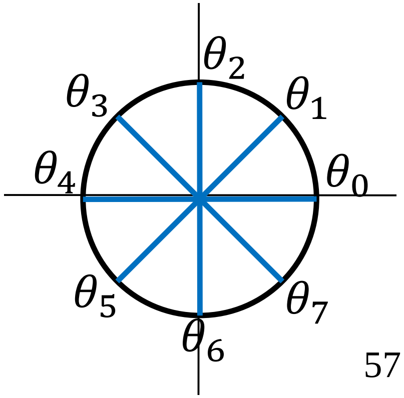
$$v_k[\cdot] = \left(e^{ik\theta_0}, \cdots, e^{ik\theta_{n-1}}\right)$$

do not have unit norm!

$$\boxed{\overline{\lambda_k}[\alpha] = \sqrt{n} \cdot v_k[\alpha]}$$

We need to normalize these functions to make them unit-norm:

$$v_k[\cdot] = \sqrt{\frac{1}{n}}\left(e^{ik\theta_0}, \cdots, e^{ik\theta_{n-1}}\right)$$



Moving Dot Products (Periodic Arrays)

Note 2

The arrays $v_k[\cdot]$ and $v_{k+n}[\cdot]$ are the same array:

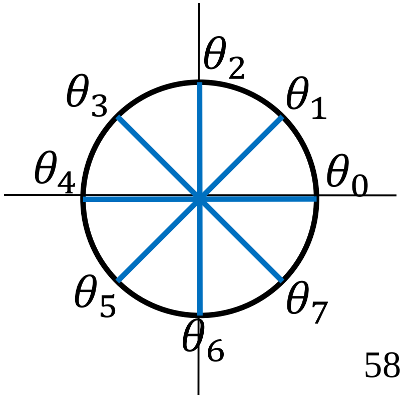
$$\begin{aligned}\sqrt{n}\cdot v_{k+n}[\cdot] &= \left(e^{i(k+n)\theta_0},\ldots,e^{i(k+n)\theta_{n-1}}\right)\\ &= \left(e^{ik\theta_0}\cdot e^{in\theta_0},\ldots,e^{ik\theta_{n-1}}\cdot e^{in\theta_{n-1}}\right)\end{aligned}$$

But $n\theta_j$ is a multiple of 2π:

$$n\theta_j = \frac{n2\pi j}{n} = 2\pi j$$

$$\Updownarrow$$

$$e^{in\theta_j} = 1$$



Moving Dot Products (Periodic Arrays)

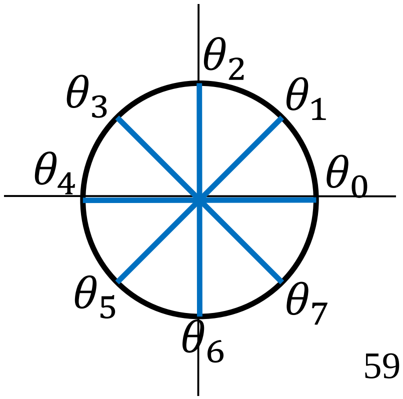
Note 2

The arrays $v_k[\cdot]$ and $v_{k+n}[\cdot]$ are the same array:

$$\begin{aligned}\sqrt{n}\cdot v_{k+n}[\cdot] &= \left(e^{i(k+n)\theta_0},\dots,e^{i(k+n)\theta_{n-1}}\right)\\ &= \left(e^{ik\theta_0}\cdot e^{in\theta_0},\dots,e^{ik\theta_{n-1}}\cdot e^{in\theta_{n-1}}\right)\end{aligned}$$

But $n\theta_j$ is a multiple of 2π so:

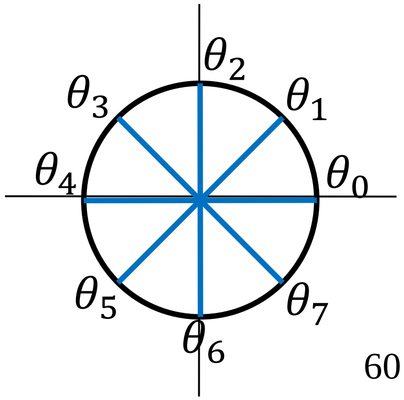
$$\begin{aligned}\sqrt{n}\cdot v_{k+n}[\cdot] &= \left(e^{ik\theta_0}\cdot e^{in\theta_0},\dots,e^{ik\theta_{n-1}}\cdot e^{in\theta_{n-1}}\right)\\ &= \left(e^{ik\theta_0},\dots,e^{ik\theta_{n-1}}\right)\\ &= \sqrt{n}\cdot v_k[\cdot]\end{aligned}$$



Moving Dot Products (Periodic Arrays)

<u>Note 3</u>

The arrays $\{v_0[\cdot], \cdots, v_{n-1}[\cdot]\}$ are linearly independent.



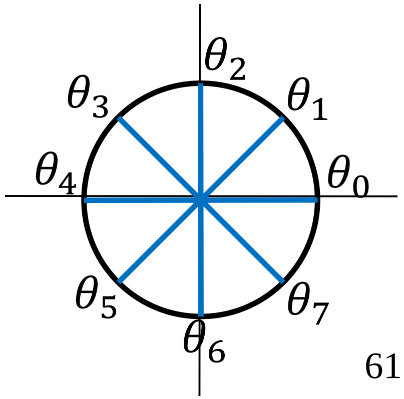
Moving Dot Products (Periodic Arrays)

Thus, given two n-dimensional arrays, $g[\cdot]$ and $h[\cdot]$, we can expand:

$$g[\cdot] = \sum_{k=0}^{n-1} \hat{g}_k \cdot v_k[\cdot] \quad \text{and} \quad h[\cdot] = \sum_{k=0}^{n-1} \hat{h}_k \cdot v_k[\cdot]$$

This gives:

$$\begin{aligned}(g[\cdot] \star h[\cdot])[\alpha] &= \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot \overline{\lambda_k}[\alpha] \\ &= \sqrt{n} \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot v_k[\alpha]\end{aligned}$$



Outline

Review

Moving Dot Products:

- One-Dimensional (Continuous)
- One-Dimensional (Discrete)
- Higher-Dimensional
- Computational Complexity

Moving Dot Products (Higher Dimensions)

The same kind of method can be used for higher dimensions:

- Periodic functions in 2D

$$f_{lm}(\theta,\phi)=\sqrt{\frac{1}{(2\pi)^2}}e^{il\theta}\cdot e^{im\phi}$$

$$\overline{\lambda_{lm}}(\alpha,\beta)=\sqrt{(2\pi)^2}f_{lm}(\alpha,\beta)$$

- Periodic functions in 3D

$$f_{lmn}(\theta,\phi,\psi)=\sqrt{\frac{1}{(2\pi)^3}}e^{il\theta}\cdot e^{im\phi}\cdot e^{in\psi}$$

$$\overline{\lambda_{lmn}}(\alpha,\beta,\gamma)=\sqrt{(2\pi)^3}f_{lmn}(\alpha,\beta,\gamma)$$

Outline

Review

Moving Dot Products:

- One-Dimensional (Continuous)
- One-Dimensional (Discrete)
- Higher-Dimensional
- Computational Complexity

Computational Complexity

What do we need to do in order to compute the moving dot-product of two periodic, n-dimensional arrays $g[\cdot]$ and $h[\cdot]$?

Computational Complexity

To compute the moving dot-product of two periodic, n-dimensional arrays $g[\cdot]$ and $h[\cdot]$:

1. We need to express $g[\cdot]$ and $h[\cdot]$ in the basis $v_k[\cdot]$:
$$g[\cdot] = \sum_{k=0}^{n-1} \hat{g}_k \cdot v_k[\cdot] \quad \text{and} \quad h[\cdot] = \sum_{k=0}^{n-1} \hat{h}_k \cdot v_k[\cdot]$$
2. We need to multiply (and scale) the coefficients:
$$(g[\cdot] \star h[\cdot])[\cdot] = \sqrt{n} \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot v_k[\cdot]$$
3. We need to evaluate at every index α:
$$(g[\cdot] \star h[\cdot])[\alpha] = \sqrt{n} \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot v_k[\alpha]$$

Computational Complexity

To compute the moving dot-product of two periodic, n-dimensional arrays $g[\cdot]$ and $h[\cdot]$:

The first and third steps are a change of bases.

These can be implemented as matrix multiplication and may be quadratic in n.

Computational Complexity

To compute the moving dot-product of two periodic, n-dimensional arrays $g[\cdot]$ and $h[\cdot]$:

1. We need to express $g[\cdot]$ and $h[\cdot]$ in the basis $v_k[\cdot]$:
$$g[\cdot] = \sum_{k=0}^{n-1} \hat{g}_k \cdot v_k[\cdot] \quad \text{and} \quad h[\cdot] = \sum_{k=0}^{n-1} \hat{h}_k \cdot v_k[\cdot] \qquad \boxed{O(N^2)}$$
2. We need to multiply (and scale) the coefficients:
$$(g[\cdot] \star h[\cdot])[\cdot] = \sqrt{n} \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot v_k[\cdot] \qquad \boxed{O(N)}$$
3. We need to evaluate at every index α:
$$(g[\cdot] \star h[\cdot])[\alpha] = \sqrt{n} \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot v_k[\alpha] \qquad \boxed{O(N^2)}$$

Computational Complexity

To compute the moving dot-product of two periodic, n-dimensional arrays $g[\cdot]$ and $h[\cdot]$:

The Fast Fourier Transform (FFT) is an algorithm for expressing an array represented by samples at $\{\theta_0, \cdots, \theta_{n-1}\}$ as a linear sum of $v_k[\cdot]$.

The Fast Inverse Fourier Transform (IFFT) is an algorithm for expressing an array represented as a linear sum of $v_k[\cdot]$ by samples at $\{\theta_0, \cdots, \theta_{n-1}\}$.

Both take $O(N \log N)$ time.

Computational Complexity

To compute the moving dot-product of two periodic, n-dimensional arrays $g[\cdot]$ and $h[\cdot]$:

1. We need to express $g[\cdot]$ and $h[\cdot]$ in the basis $v_k[\cdot]$:
$$g[\cdot] = \sum_{k=0}^{n-1} \hat{g}_k \cdot v_k[\cdot] \quad \text{and} \quad h[\cdot] = \sum_{k=0}^{n-1} \hat{h}_k \cdot v_k[\cdot]$$
$\boxed{O(N \log N)}$
2. We need to multiply (and scale) the coefficients:
$$(g[\cdot] \star h[\cdot])[\cdot] = \sqrt{n} \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot v_k[\cdot]$$
$\boxed{O(N)}$
3. We need to evaluate at every index α:
$$(g[\cdot] \star h[\cdot])[\alpha] = \sqrt{n} \sum_{k=0}^{n-1} \hat{g}_k \cdot \overline{\hat{h}_k} \cdot v_k[\alpha]$$
$\boxed{O(N \log N)}$

The Inverse Fourier Transform

The Fourier Transform is a change of basis transformation:

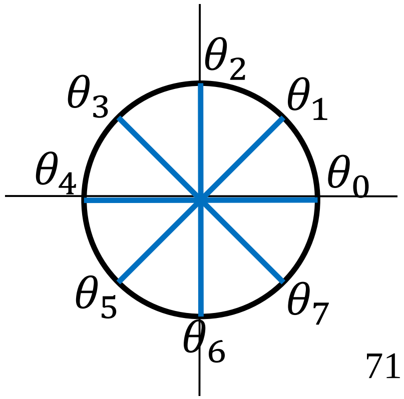
Evaluation Basis

$$
\begin{gathered}
(1,0,\cdots,0,0)\\
(0,1,\cdots,0,0)\\
\vdots\\
(0,0,\cdots,1,0)\\
(0,0,\cdots,0,1)
\end{gathered}
$$

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

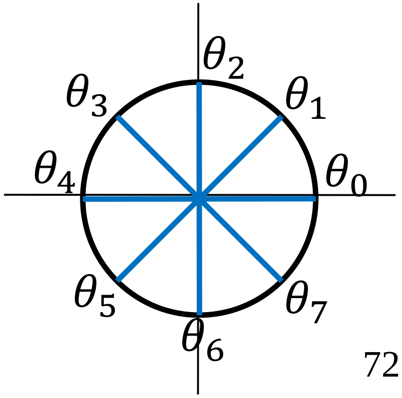
Complex Exponential Basis

$$
\begin{gathered}
\left(e^{i0\theta_0},e^{i0\theta_1},\dots,e^{i0\theta_{n-2}},e^{i0\theta_{n-1}}\right)\\
\left(e^{i1\theta_0},e^{i1\theta_1},\dots,e^{i1\theta_{n-2}},e^{i1\theta_{n-1}}\right)\\
\vdots\\
\left(e^{i(n-2)\theta_0},e^{i(n-2)\theta_1},\dots,e^{i(n-2)\theta_{n-2}},e^{i(n-2)\theta_{n-1}}\right)\\
\left(e^{i(n-1)\theta_0},e^{i(n-1)\theta_1},\dots,e^{i(n-1)\theta_{n-2}},e^{i(n-1)\theta_{n-1}}\right)
\end{gathered}
$$



The Inverse Fourier Transform

Since the Fourier basis is orthonormal, we can get the k-th Fourier coefficient by taking the dot-product with the k-th basis function.



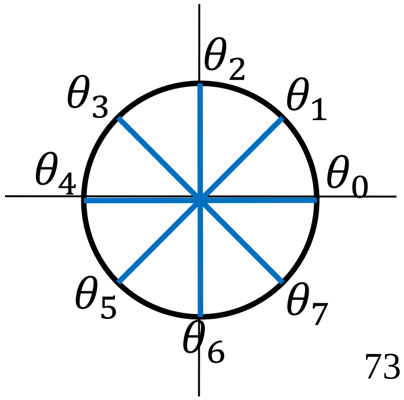
The Inverse Fourier Transform

This can be represented by the matrix:

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

Where θ is the angle:

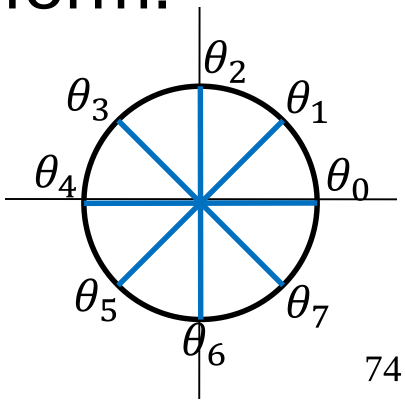
$$\theta = \frac{2\pi}{n}$$



The Inverse Fourier Transform

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

Since both bases are orthogonal, the matrix is unitary, and the inverse Fourier transform is the transpose conjugate of the forward transform.



The Inverse Fourier Transform

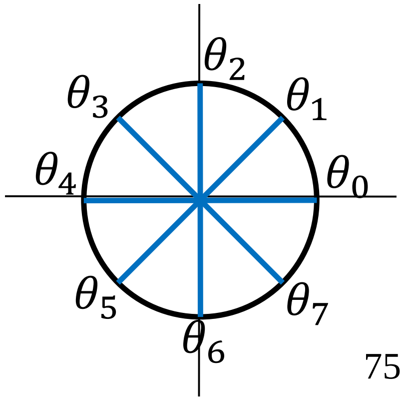
$$F = \sqrt{\frac{1}{n}}\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & e^{-i\theta} & \cdots & e^{-i(n-2)\theta} & e^{-i(n-1)\theta} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & e^{-i(n-2)\theta} & \cdots & e^{-i(n-2)(n-2)\theta} & e^{-i(n-2)(n-1)\theta} \\ 1 & e^{-i(n-1)\theta} & \cdots & e^{-i(n-1)(n-2)\theta}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

In particular, given the Fourier coefficients:

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

The inverse Fourier transform is:

$$F^{-1}\begin{pmatrix} \hat{a}_0 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix} = \bar{F}^t \begin{pmatrix} \hat{a}_0 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix}$$

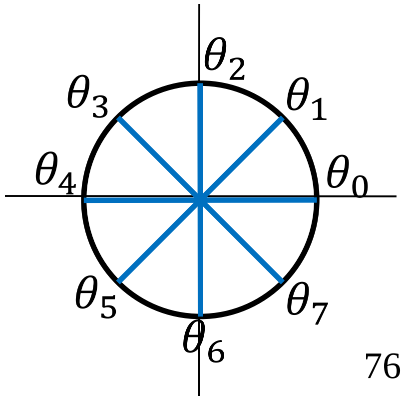


The Inverse Fourier Transform

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

Taking the double conjugate, we get:

$$F^{-1}\begin{pmatrix} \hat{a}_0 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix} = \overline{\overline{\bar{F}^t\begin{pmatrix} \hat{a}_0 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix}}} = F^t\begin{pmatrix} \overline{\hat{a}_0} \\ \vdots \\ \overline{\hat{a}_{n-1}} \end{pmatrix}$$

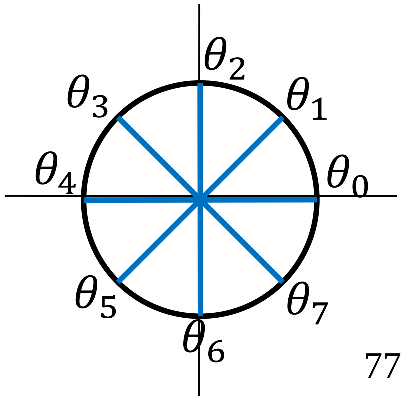


The Inverse Fourier Transform

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

Since $F=F^t$, this gives:

$$F^{-1}\begin{pmatrix} \hat{a}_0 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix} = \overline{F\begin{pmatrix} \overline{\hat{a}_0} \\ \vdots \\ \overline{\hat{a}_{n-1}} \end{pmatrix}}$$



The Inverse Fourier Transform

$$F^{-1}\begin{pmatrix}\hat{a}_0\\ \vdots\\ \hat{a}_{n-1}\end{pmatrix} = \overline{F\begin{pmatrix}\overline{\hat{a}_0}\\ \vdots\\ \overline{\hat{a}_{n-1}}\end{pmatrix}}$$

We can compute the inverse transform by:

1. Taking the conjugate of the Fourier coefficients
2. Computing the forward Fourier transform
3. Taking the conjugate of the resultant coefficients.