Advanced Algorithms

COMP4121

Aleks Ignjatovic

School of Computer Science and Engineering
University of New South Wales

The Discrete Fourier Transform, Discrete Cosine Transform, Convolution and their Applications

The DFT as a change of basis

- Recall that the *scalar product* (also called the *dot* product) of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$, denoted by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$, is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i y_i.$$

- If the coordinates of the two vectors are complex numbers, i.e., if $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^n$, then the scalar product is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i \overline{y_i} \tag{1}$$

- $\overline{z}$ denotes the complex conjugate of z, i.e., if $z = a + i\,b$ where $a, b \in \mathbb{R}$, then $\overline{z} = a - i\,b$.
- Note that $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$.

The DFT as a change of basis

- Recall that the *scalar product* (also called the *dot* product) of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$, denoted by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$, is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i y_i.$$

- If the coordinates of the two vectors are complex numbers, i.e., if $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^n$, then the scalar product is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i \overline{y_i}. \tag{1}$$

- $\overline{z}$ denotes the complex conjugate of z, i.e., if $z = a + i\,b$ where $a, b \in \mathbb{R}$, then $\overline{z} = a - i\,b$.
- Note that $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$.

The DFT as a change of basis

- Recall that the *scalar product* (also called the *dot* product) of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$, denoted by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$, is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i y_i.$$

- If the coordinates of the two vectors are complex numbers, i.e., if $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^n$, then the scalar product is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i \overline{y_i}. \tag{1}$$

- $\overline{z}$ denotes the complex conjugate of z, i.e., if $z = a + i\,b$ where $a, b \in \mathbb{R}$, then $\overline{z} = a - i\,b$.

- Note that $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$.

The DFT as a change of basis

- Recall that the *scalar product* (also called the *dot* product) of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$, denoted by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$, is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i y_i.$$

- If the coordinates of the two vectors are complex numbers, i.e., if $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{C}^n$, then the scalar product is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=0}^{n-1} x_i \overline{y_i}. \tag{1}$$

- $\overline{z}$ denotes the complex conjugate of z, i.e., if $z = a + i\,b$ where $a, b \in \mathbb{R}$, then $\overline{z} = a - i\,b$.

- Note that $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$.

- In fact, in a vector spa… binary function $\langle \boldsymbol{x}, \boldsymbol{y} \rangle : V^2 \rightarrow \mathbb{C}$ which satisfies the following three pro… scalar product because it has all of the important properties w… articular scalar product given by (1) has.
  1. (Conjugate symmetry) $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$
  2. (Linearity in the first argument) for every scalar α, $\langle \alpha \boldsymbol{x}, \boldsymbol{y} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
  3. (Positive definitness) $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0$ just in case $\boldsymbol{x} = 0$
- Every scalar product also defines an associated *norm* of a vector by

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \geq 0$$

- Geometrically, the norm always the length of a vector (and, in case of $\mathbb{R}^2$ or $\mathbb{R}^3$, it is just the usual, Euclidean length of the vector).

- In fact, in a vector spa[illegible] binary function $\langle \boldsymbol{x}, \boldsymbol{y} \rangle : V^2 \to \mathbb{C}$ which satisfies the following three pro[illegible] scalar product because it has all of the important properties w[illegible] articular scalar product given by (1) has.
  1. (Conjugate symmetry) $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$
  2. (Linearity in the first argument) for every scalar α, $\langle \alpha \boldsymbol{x}, \boldsymbol{y} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
  3. (Positive definitness) $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0$ just in case $\boldsymbol{x} = 0$
- Every scalar product also defines an associated *norm* of a vector by

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \geq 0$$

- Geometrically, the norm [illegible] length of a vector (and, in case of $\mathbb{R}^2$ or $\mathbb{R}^3$, it is just the usual, Euclidean length of the vector).

- In fact, in a vector spa[illegible]binary function $\langle \boldsymbol{x}, \boldsymbol{y} \rangle : V^2 \to \mathbb{C}$ which satisfies the following three pro[illegible] scalar product because it has all of the important properties w[illegible]articular scalar product given by (1) has.
  1. (Conjugate symmetry) $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$
  2. (Linearity in the first argument) for every scalar α, $\langle \alpha \boldsymbol{x}, \boldsymbol{y} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
  3. (Positive definitness) $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0$ just in case $\boldsymbol{x} = 0$
- Every scalar product also defines an associated *norm* of a vector by

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \geq 0$$

- Geometrically, the norm plays the role of the length of a vector (and, in case of $\mathbb{R}^2$ or $\mathbb{R}^3$, it is just the usual, Euclidean length of the vector).

- In fact, in a vector spa[illegible]binary function $\langle \boldsymbol{x}, \boldsymbol{y} \rangle : V^2 \rightarrow \mathbb{C}$ which satisfies the following three pro[illegible] scalar product because it has all of the important properties w[illegible]articular scalar product given by (1) has.
  1. (Conjugate sym[illegible]) $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$
  2. (Linearity in the first argument) for every scalar α, $\langle \alpha\boldsymbol{x}, \boldsymbol{y} \rangle = \alpha\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
  3. (Positive definitness) $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0$ just in case $\boldsymbol{x} = 0$

- Every scalar product also defines an associated *norm* of a vector by

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \geq 0$$

- Geometrically, the norm plays the role of the length of a vector (and, in case of $\mathbb{R}^2$ or $\mathbb{R}^3$, it is just the usual, Euclidean length of the vector).

The DFT as a change of basis

- In fact, in a vector spa[illegible]binary function $\langle \boldsymbol{x}, \boldsymbol{y} \rangle : V^2 \rightarrow \mathbb{C}$ which satisfies the following three pro[illegible] scalar product because it has all of the important properties w[illegible]articular scalar product given by (1) has.
  1. (Conjugate symmetry) $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$
  2. (Linearity in the first argument) for every scalar α, $\langle \alpha \boldsymbol{x}, \boldsymbol{y} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
  3. (Positive definitness) $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0$ just in case $\boldsymbol{x} = 0$
- Every scalar product also defines an associated *norm* of a vector by

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \geq 0$$

- Geometrically, the norm plays the role of the length of a vector (and, in case of $\mathbb{R}^2$ or $\mathbb{R}^3$, it is just the usual, Euclidean length of the vector).

The DFT as a change of basis

- In fact, in a vector space V, any binary function $\langle \boldsymbol{x}, \boldsymbol{y} \rangle : V^2 \to \mathbb{C}$ which satisfies the following three properties is called a scalar product because it has all of the important properties which the particular scalar product given by (1) has.
  1. (Conjugate symmetry) $\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$
  2. (Linearity in the first argument) for every scalar α, $\langle \alpha \boldsymbol{x}, \boldsymbol{y} \rangle = \alpha \langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{x} + \boldsymbol{y}, \boldsymbol{z} \rangle = \langle \boldsymbol{x}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z} \rangle$
  3. (Positive definitness) $\langle \boldsymbol{x}, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = 0$ just in case $\boldsymbol{x} = 0$

- Every scalar product also defines an associated *norm* of a vector by

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \geq 0$$

- Geometrically, the norm plays the role of the length of a vector (and, in case of $\mathbb{R}^2$ or $\mathbb{R}^3$, it is just the usual, Euclidean length of the vector).

The DFT as a change of basis

- It can be shown that the axioms for the scalar product imply that a scalar product and its corresponding norm satisfy the Cauchy-Schwarz inequality:

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|.$$

- If the field of scalars is the field of real numbers $\mathbb{R}$, then the scalar product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ of any two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is a real number and the Cauchy - Schwarz inequality allows us to define angles between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ so that

$$\cos \angle(\boldsymbol{u}, \boldsymbol{v}) = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}.$$

- If the field of scalars is the field of complex numbers $\mathbb{C}$, then the angle between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$\cos \angle(\boldsymbol{u}, \boldsymbol{v}) = \frac{\Re\mathfrak{e}\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}.$$

The DFT as a change of basis

- It can be shown that the axioms for the scalar product imply that a scalar product and its corresponding norm satisfy the Cauchy-Schwarz inequality:

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|.$$

- If the field of scalars is the field of real numbers $\mathbb{R}$, then the scalar product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ of any two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is a real number and the Cauchy - Schwarz inequality allows us to define angles between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ so that

$$\cos \angle(\boldsymbol{u}, \boldsymbol{v}) = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}.$$

- If the field of scalars is the field of complex numbers $\mathbb{C}$, then the angle between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$\cos \angle(\boldsymbol{u}, \boldsymbol{v}) = \frac{\mathfrak{Re}\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}.$$

The DFT as a change of basis

- It can be shown that the axioms for the scalar product imply that a scalar product and its corresponding norm satisfy the Cauchy-Schwarz inequality:

$$|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|.$$

- If the field of scalars is the field of real numbers $\mathbb{R}$, then the scalar product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ of any two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is a real number and the Cauchy - Schwarz inequality allows us to define angles between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ so that

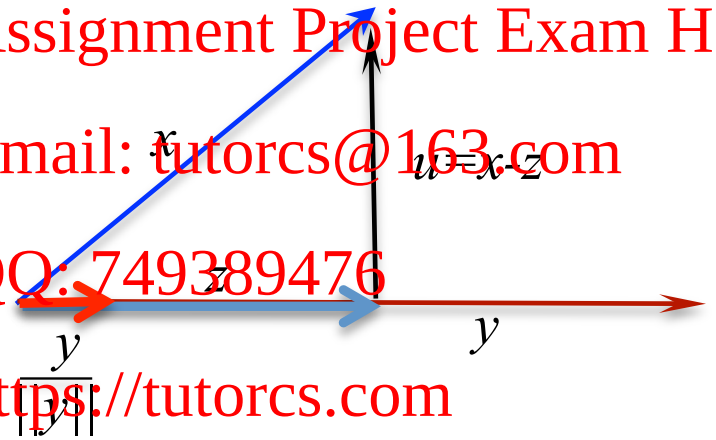
$$\cos \angle(\boldsymbol{u}, \boldsymbol{v}) = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}.$$

- If the field of scalars is the field of complex numbers $\mathbb{C}$, then the angle between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$\cos \angle(\boldsymbol{u}, \boldsymbol{v}) = \frac{\mathfrak{Re}\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}.$$

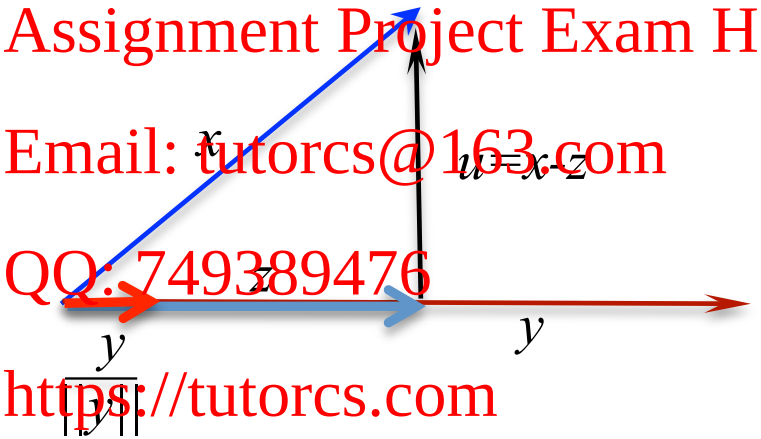
The DFT as a change of basis

- Thus, if the field of sca[illegible]set of reals $\mathbb{R}$, since we have a length function (the norm $\|x\|$ of a vec[illegible]n angle function, such a vector space with a scalar product has a w[illegible]geometry.
- If $\boldsymbol{y}$ is a unit vector, $\|\boldsymbol{y}\| = 1$, then we have $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \|x\| \cdot \cos(\angle \boldsymbol{x}, \boldsymbol{y})$, i.e., $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is just the length of the orthogonal projection of $\boldsymbol{x}$ onto the line to which $\boldsymbol{y}$ belongs; see the figure.



The DFT as a change of basis

- Thus, if the field of sca[illegible]set of reals $\mathbb{R}$, since we have a length function (the norm $\|x\|$ of a vec[illegible]n angle function, such a vector space with a scalar product has a w[illegible]geometry.
- If $\boldsymbol{y}$ is a unit vector, $\|\boldsymbol{y}\| = 1$, then we have $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \|x\| \cdot \cos(\angle \boldsymbol{x}, \boldsymbol{y})$, i.e., $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is just the length of the orthogonal projection of $\boldsymbol{x}$ onto the line to which $\boldsymbol{y}$ belongs; see the figure.

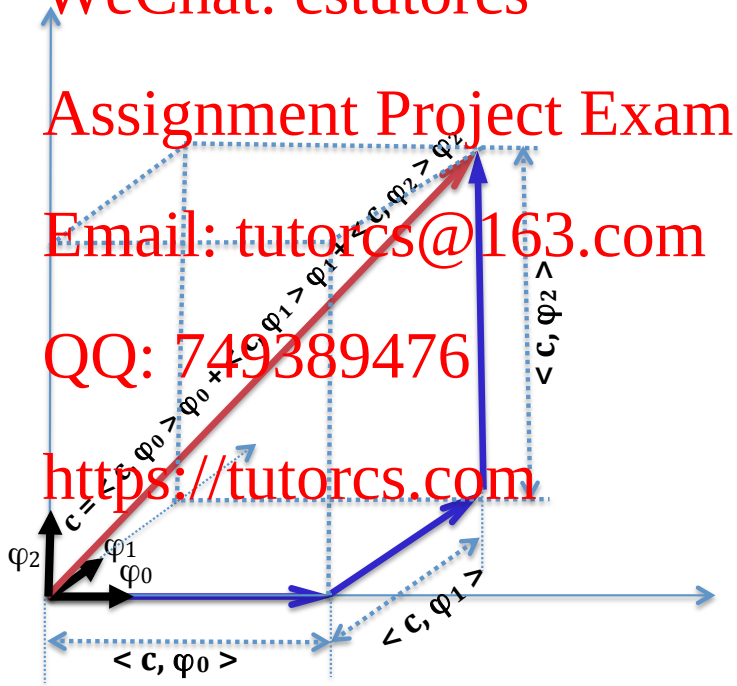


The DFT as a change of basis

Theorem

Let $\{\boldsymbol{\varphi}_m\}_{0\le m<n}$ *be any orthonormal basis of* $\mathbb{C}^n$ *and* $\boldsymbol{c}$ *any vector in* $\mathbb{C}^n$*; then*

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle \boldsymbol{\varphi}_m; \tag{2}$$



The DFT as a change of basis

- The usual basis of both $\mathbb{R}^n$ and $\mathbb{C}^n$ is given by

$$\mathcal{B} = \{(1, 0, 0, \ldots, 0), (0, 1, 0, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, 1)\}$$

- Note that for any vector $(a_0, a_1, a_2, \ldots, a_{n-1})$, such that $\boldsymbol{a} \in \mathbb{R}^n$ or $\boldsymbol{a} \in \mathbb{C}^n$,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0(1, 0, 0, \ldots, 0) + \ldots + a_{n-1}(0, 0, 0, \ldots, 1)$$

- Let us set

$$\boldsymbol{e}_0 = (1, 0, 0, \ldots, 0, 0); \quad \boldsymbol{e}_1 = (0, 1, 0, \ldots, 0, 0); \quad \cdots \quad \boldsymbol{e}_{n-1} = (0, 0, 0, \ldots, 0, 1);$$

- Thus,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0\boldsymbol{e_0} + a_1\boldsymbol{e}_1 + \ldots + a_{n-1}\boldsymbol{e}_{n-1} = \sum_{i=0}^{n-1} a_i\boldsymbol{e}_i$$

The DFT as a change of basis

- The usual basis of both $\mathbb{R}^n$ and $\mathbb{C}^n$ is given by

$$\mathcal{B} = \{(1, 0, 0, \ldots, 0), (0, 1, 0, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, 1)\}$$

- Note that for any vector $(a_0, a_1, a_2, \ldots, a_{n-1})$, such that $\boldsymbol{a} \in \mathbb{R}^n$ or $\boldsymbol{a} \in \mathbb{C}^n$,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0(1, 0, 0, 0, \ldots, 0) + \ldots + a_{n-1}(0, 0, 0, \ldots, 1)$$

- Let us set

$$\boldsymbol{e}_0 = (1, 0, 0, \ldots, 0, 0); \quad \boldsymbol{e}_1 = (0, 1, 0, \ldots, 0, 0); \quad \cdots \quad \boldsymbol{e}_{n-1} = (0, 0, 0, \ldots, 0, 1);$$

- Thus,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0\boldsymbol{e}_0 + a_1\boldsymbol{e}_1 + \ldots + a_{n-1}\boldsymbol{e}_{n-1} = \sum_{i=0}^{n-1} a_i\boldsymbol{e}_i$$

The DFT as a change of basis

- The usual basis of both $\mathbb{R}^n$ and $\mathbb{C}^n$ is given by

$$\mathcal{B} = \{(1, 0, 0, \ldots, 0), (0, 1, 0, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, 1)\}$$

- Note that for any vector $(a_0, a_1, a_2, \ldots, a_{n-1})$, such that $\boldsymbol{a} \in \mathbb{R}^n$ or $\boldsymbol{a} \in \mathbb{C}^n$,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0(1, 0, 0, 0, \ldots, 0) + \ldots + a_{n-1}(0, 0, 0, \ldots, 1)$$

- Let us set

$$\boldsymbol{e}_0 = (1, 0, 0, \ldots, 0, 0); \quad \boldsymbol{e}_1 = (0, 1, 0, \ldots, 0, 0); \quad \ldots \quad \boldsymbol{e}_{n-1} = (0, 0, 0, \ldots, 0, 1);$$

- Thus,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0\boldsymbol{e}_0 + a_1\boldsymbol{e}_1 + \ldots + a_{n-1}\boldsymbol{e}_{n-1} = \sum_{i=0}^{n-1} a_i\boldsymbol{e}_i$$

The DFT as a change of basis

- The usual basis of both $\mathbb{R}^n$ and $\mathbb{C}^n$ is given by

$$\mathcal{B} = \{(1, 0, 0, \ldots, 0), (0, 1, 0, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, 1)\}$$

- Note that for any vector $(a_0, a_1, a_2, \ldots, a_{n-1})$, such that $\boldsymbol{a} \in \mathbb{R}^n$ or $\boldsymbol{a} \in \mathbb{C}^n$,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0(1, 0, 0, \ldots, 0) + \ldots + a_{n-1}(0, 0, 0, \ldots, 1)$$

- Let us set

$$\boldsymbol{e}_0 = (1, 0, 0, \ldots, 0, 0); \quad \boldsymbol{e}_1 = (0, 1, 0, \ldots, 0, 0); \quad \ldots \quad \boldsymbol{e}_{n-1} = (0, 0, 0, \ldots, 0, 1);$$

- Thus,

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0\boldsymbol{e_0} + a_1\boldsymbol{e}_1 + \ldots + a_{n-1}\boldsymbol{e}_{n-1} = \sum_{i=0}^{n-1} a_i\boldsymbol{e}_i$$

The DFT as a change of basis

- Let us denote the complex number $e^{i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ by ω_n.
- Such a number is a *primitive root of unity* because $(\omega_n)^n = 1$, and the set of all complex numbers z satisfying $z^n = 1$ is precisely the set of the n powers of ω_n.
- Thus, $z^n = 1$ if an only if z is of the form $z = (\omega_n)^k = \omega_n^k$ for some k such that $0 \leq k \leq n-1$.
- We now introduce another basis, this time only in $\mathbb{C}^n$, given by $\mathcal{F} = \{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{n-1}\}$, where

$$\boldsymbol{f}_k = ((\omega_n^k)^0, (\omega_n^k)^1, \ldots, (\omega_n^k)^{n-1}) = (1, \omega_n^k, \omega_n^{2k}, \ldots, \omega_n^{k(n-1)}).$$

- Thus, the coordinates of $\boldsymbol{f}_k$ are the sequence of n consecutive powers of ω_n^k.
- To show that this is indeed a basis we have to show that these vectors are linearly independent.
- In fact, they form an orthogonal basis. Two vectors are mutually orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

The DFT as a change of basis

- Let us denote the complex number $e^{i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ by ω_n.
- Such a number is a *pr* *of unity* because $(\omega_n)^n = 1$, and the set of all complex numbers z sa = 1 is precisely the set of the n powers of ω_n.
- Thus, $z^n = 1$ if an onl he form $z = (\omega_n)^k = \omega_n^k$ for some k such that $0 \le k \le n-1$.
- We now introduce another basis, this time only in $\mathbb{C}^n$, given by $\mathcal{F} = \{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{n-1}\}$, where

$$\boldsymbol{f}_k = ((\omega_n^k)^0, (\omega_n)^{\quad}, \ldots, (\omega_n)^{\quad}) = (1, \omega_n, \omega_n^{2\quad}, \ldots, \omega_n^{k(n-1)}).$$

- Thus, the coordinates of $\boldsymbol{f}_k$ are the sequence of n consecutive powers of ω_n^k.
- To show that this is indeed a basis we have to show that these vectors are linearly independent.
- In fact, they form an orthogonal basis. Two vectors are mutually orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

The DFT as a change of basis

- Let us denote the complex number $e^{i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ by ω_n.
- Such a number is a *pr* *of unity* because $(\omega_n)^n = 1$, and the set of all complex numbers z sa = 1 is precisely the set of the n powers of ω_n.
- Thus, $z^n = 1$ if an onl he form $z = (\omega_n)^k = \omega_n^k$ for some k such that $0 \le k \le n-1$.
- We now introduce another basis, this time only in $\mathbb{C}^n$, given by $\mathcal{F} = \{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{n-1}\}$, where

$$\boldsymbol{f}_k = ((\omega_n^k)^0, (\omega_n)^{2\cdot}, \ldots, (\omega_n)^{\cdot\cdot}) = (1, \omega_n, \omega_n^{2\cdot}, \ldots, \omega_n^{k(n-1)}).$$

- Thus, the coordinates of $\boldsymbol{f}_k$ are the sequence of n consecutive powers of ω_n^k.
- To show that this is indeed a basis we have to show that these vectors are linearly independent.
- In fact, they form an orthogonal basis. Two vectors are mutually orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

The DFT as a change of basis

- Let us denote the complex number $e^{i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ by ω_n.
- Such a number is a *pr* *of unity* because $(\omega_n)^n = 1$, and the set of all complex numbers z sa = 1 is precisely the set of the n powers of ω_n.
- Thus, $z^n = 1$ if an onl the form $z = (\omega_n)^k = \omega_n^k$ for some k such that $0 \le k \le n-1$.
- We now introduce another basis, this time only in $\mathbb{C}^n$, given by $\mathcal{F} = \{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{n-1}\}$, where

$$\boldsymbol{f}_k = ((\omega_n^k)^0, (\omega_n^k)^1, \ldots, (\omega_n^k)^{n-1}) = (1, \omega_n^k, \omega_n^{2k}, \ldots, \omega_n^{k(n-1)}).$$

- Thus, the coordinates of $\boldsymbol{f}_k$ are the sequence of n consecutive powers of ω_n^k.
- To show that this is indeed a basis we have to show that these vectors are linearly independent.
- In fact, they form an orthogonal basis. Two vectors are mutually orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

The DFT as a change of basis

- Let us denote the complex number $e^{i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ by ω_n.
- Such a number is a *pr* *of unity* because $(\omega_n)^n = 1$, and the set of all complex numbers z sa = 1 is precisely the set of the n powers of ω_n.
- Thus, $z^n = 1$ if an onl the form $z = (\omega_n)^k = \omega_n^k$ for some k such that $0 \le k \le n-1$.
- We now introduce another basis, this time only in $\mathbb{C}^n$, given by $\mathcal{F} = \{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{n-1}\}$, where

$$\boldsymbol{f}_k = ((\omega_n^k)^0, (\omega_n^k)^1, \ldots, (\omega_n^k)^{n-1}) = (1, \omega_n^k, \omega_n^{k2}, \ldots, \omega_n^{k(n-1)}).$$

- Thus, the coordinates of $\boldsymbol{f}_k$ are the sequence of n consecutive powers of ω_n^k.
- To show that this is indeed a basis we have to show that these vectors are linearly independent.
- In fact, they form an orthogonal basis. Two vectors are mutually orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

The DFT as a change of basis

- Let us denote the complex number $e^{i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ by ω_n.
- Such a number is a *pr* *of unity* because $(\omega_n)^n = 1$, and the set of all complex numbers z sa $= 1$ is precisely the set of the n powers of ω_n.
- Thus, $z^n = 1$ if an onl the form $z = (\omega_n)^k = \omega_n^k$ for some k such that $0 \le k \le n-1$.
- We now introduce another basis, this time only in $\mathbb{C}^n$, given by $\mathcal{F} = \{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{n-1}\}$, where

$$\boldsymbol{f}_k = ((\omega_n^k)^0, (\omega_n^k)^1, \ldots, (\omega_n^k)^{n-1}) = (1, \omega_n^k, \omega_n^{2k}, \ldots, \omega_n^{k(n-1)}).$$

- Thus, the coordinates of $\boldsymbol{f}_k$ are the sequence of n consecutive powers of ω_n^k.
- To show that this is indeed a basis we have to show that these vectors are linearly independent.
- In fact, they form an orthogonal basis. Two vectors are mutually orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

The DFT as a change of basis

- Let us denote the complex number $e^{i\frac{2\pi}{n}} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$ by ω_n.
- Such a number is a *pr* *of unity* because $(\omega_n)^n = 1$, and the set of all complex numbers z sa = 1 is precisely the set of the n powers of ω_n.
- Thus, $z^n = 1$ if an onl the form $z = (\omega_n)^k = \omega_n^k$ for some k such that $0 \le k \le n-1$.
- We now introduce another basis, this time only in $\mathbb{C}^n$, given by $\mathcal{F} = \{\boldsymbol{f}_0, \ldots, \boldsymbol{f}_{n-1}\}$, where

$$\boldsymbol{f}_k = ((\omega_n^k)^0, (\omega_n^k)^1, \ldots, (\omega_n^k)^{n-1}) = (1, \omega_n^k, \omega_n^{2k}, \ldots, \omega_n^{k(n-1)}).$$

- Thus, the coordinates of $\boldsymbol{f}_k$ are the sequence of n consecutive powers of ω_n^k.
- To show that this is indeed a basis we have to show that these vectors are linearly independent.
- In fact, they form an orthogonal basis. Two vectors are mutually orthogonal if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = 0$.

The DFT as a change of basis

- To see that vectors $\boldsymbol{f}_k$ and $\boldsymbol{f}_m$ are orthogonal if $k \neq m$ we compute their scalar product:

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{m \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-m \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^{(k-m) \cdot i} = \sum_{i=0}^{n-1} (\omega_n^{k-m})^i$$

- If $k \neq m$ the last sum is a sum of a geometric progression with the ratio $q = \omega_n^{k-m}$
- Thus, using formula

$$\sum_{i=0}^{n-1} q^i = \frac{1 - q^n}{1 - q}$$

we get

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \frac{1 - (\omega_n^{k-m})^n}{1 - \omega_n^{k-m}} = \frac{1 - (\omega_n)^{k-m \cdot n}}{1 - \omega_n^{k-m}} = \frac{1 - 1^{k-m}}{1 - \omega_n^{k-m}} = 0$$

(the denominator is different from 0 because we have assumed that $k \neq m$).

- Let us compute the norm of these vectors:

$$\|\boldsymbol{f}_k\|^2 = \langle \boldsymbol{f}_k, \boldsymbol{f}_k \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{k \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-k \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^0 = \sum_{i=0}^{n-1} 1 = n.$$

- Thus, $\|\boldsymbol{f}_k\| = \sqrt{n}$.

The DFT as a change of basis

- To see that vectors $\boldsymbol{f}_k$ and $\boldsymbol{f}_m$ are orthogonal if $k \neq m$ we compute their scalar product:

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{m \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-m \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^{(k-m) \cdot i} = \sum_{i=0}^{n-1} (\omega_n^{k-m})^i$$

- If $k \neq m$ the last sum is a sum of a geometric progression with the ratio $q = \omega_n^{k-m}$
- Thus, using formula

$$\sum_{i=0}^{n-1} q^i = \frac{1 - q^n}{1 - q}$$

we get

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \frac{1 - (\omega_n^{k-m})^n}{1 - \omega_n^{k-m}} = \frac{1 - (\omega_n)^{k-m \cdot n}}{1 - \omega_n^{k-m}} = \frac{1 - 1^{k-m}}{1 - \omega_n^{k-m}} = 0$$

(the denominator is different from 0 because we have assumed that $k \neq m$).

- Let us compute the norm of these vectors:

$$\|\boldsymbol{f}_k\|^2 = \langle \boldsymbol{f}_k, \boldsymbol{f}_k \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{k \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-k \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^0 = \sum_{i=0}^{n-1} 1 = n.$$

- Thus, $\|\boldsymbol{f}_k\| = \sqrt{n}$.

The DFT as a change of basis

- To see that vectors $\boldsymbol{f}_k$ and $\boldsymbol{f}_m$ are orthogonal if $k \neq m$ we compute their scalar product:

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{m \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-m \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^{(k-m) \cdot i} = \sum_{i=0}^{n-1} (\omega_n^{k-m})^i$$

- If $k \neq m$ the last sum is a sum of a geometric progression with the ratio $q = \omega_n^{k-m}$
- Thus, using formula

$$\sum_{i=0}^{n-1} q^i = \frac{1-q^n}{1-q}$$

we get

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \frac{1-(\omega_n^{k-m})^n}{1-\omega_n^{k-m}} = \frac{1-(\omega_n)^{k-m}}{1-\omega_n^{k-m}} = \frac{1-1^{k-m}}{1-\omega_n^{k-m}} = 0$$

(the denominator is different from 0 because we have assumed that $k \neq m$).

- Let us compute the norm of these vectors:

$$\|\boldsymbol{f}_k\|^2 = \langle \boldsymbol{f}_k, \boldsymbol{f}_k \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{k \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-k \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^0 = \sum_{i=0}^{n-1} 1 = n.$$

- Thus, $\|\boldsymbol{f}_k\| = \sqrt{n}$.

The DFT as a change of basis

- To see that vectors $\boldsymbol{f}_k$ and $\boldsymbol{f}_m$ are orthogonal if $k \neq m$ we compute their scalar product:

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{m \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-m \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^{(k-m) \cdot i} = \sum_{i=0}^{n-1} (\omega_n^{k-m})^i$$

- If $k \neq m$ the last sum is a sum of a geometric progression with the ratio $q = \omega_n^{k-m}$
- Thus, using formula

$$\sum_{i=0}^{n-1} q^i = \frac{1-q^n}{1-q}$$

we get

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \frac{1-(\omega_n^{k-m})^n}{1-\omega_n^{k-m}} = \frac{1-(\omega_n)^{k-m}}{1-\omega_n^{k-m}} = \frac{1-1^{k-m}}{1-\omega_n^{k-m}} = 0$$

(the denominator is different from 0 because we have assumed that $k \neq m$).

- Let us compute the norm of these vectors:

$$\|\boldsymbol{f}_k\|^2 = \langle \boldsymbol{f}_k, \boldsymbol{f}_k \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{k \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-k \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^0 = \sum_{i=0}^{n-1} 1 = n.$$

- Thus, $\|\boldsymbol{f}_k\| = \sqrt{n}$.

The DFT as a change of basis

- To see that vectors $\boldsymbol{f}_k$ and $\boldsymbol{f}_m$ are orthogonal if $k \neq m$ we compute their scalar product:

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{m \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-m \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^{(k-m) \cdot i} = \sum_{i=0}^{n-1} (\omega_n^{k-m})^i$$

- If $k \neq m$ the last sum is a sum of a geometric progression with the ratio $q = \omega_n^{k-m}$
- Thus, using formula

$$\sum_{i=0}^{n-1} q^i = \frac{1-q^n}{1-q}$$

we get

$$\langle \boldsymbol{f}_k, \boldsymbol{f}_m \rangle = \frac{1-(\omega_n^{k-m})^n}{1-\omega_n^{k-m}} = \frac{1-(\omega_n)^{k-m}}{1-\omega_n^{k-m}} = \frac{1-1^{k-m}}{1-\omega_n^{k-m}} = 0$$

(the denominator is different from 0 because we have assumed that $k \neq m$).

- Let us compute the norm of these vectors:

$$\|\boldsymbol{f}_k\|^2 = \langle \boldsymbol{f}_k, \boldsymbol{f}_k \rangle = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} \overline{(\omega_n)^{k \cdot i}} = \sum_{i=0}^{n-1} (\omega_n)^{k \cdot i} (\omega_n)^{-k \cdot i} = \sum_{i=0}^{n-1} (\omega_n)^0 = \sum_{i=0}^{n-1} 1 = n.$$

- Thus, $\|\boldsymbol{f}_k\| = \sqrt{n}$.

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.
- Let us now calculate the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ of a vector $\boldsymbol{c}$ in basis $\boldsymbol{\Phi}$.
- Since $\boldsymbol{\varphi}_m = \frac{1}{\sqrt{n}} \langle e^{i \frac{2\pi}{n} m k} : 0 \leq k < n \rangle$ using the definition of the scalar product in $\mathbb{C}^n$ we have

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} m k}$$

- Vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ is called the **Discrete Fourier Transform (DFT)** of the sequence (vector) $\boldsymbol{c}$.
- Thus, the DFT of a vector $\boldsymbol{c}$ is just the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of the coordinates of $\boldsymbol{c}$ in the orthonormal basis $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$:

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.

- Let us now calculate the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ of a vector $\boldsymbol{c}$ in basis $\boldsymbol{\Phi}$.

- Since $\boldsymbol{\varphi}_m = \frac{1}{\sqrt{n}}\langle e^{i\,\frac{2\pi}{n}\,m\,0}, e^{i\,\frac{2\pi}{n}\,m\,1}, \ldots, e^{i\,\frac{2\pi}{n}\,m\,(n-1)} \rangle$ and $0 \leq k < n$ using the definition of the scalar product in $\mathbb{C}^n$ we have

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k\, e^{-i\,\frac{2\pi}{n}\,m\,k}$$

- Vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ is called the **Discrete Fourier Transform (DFT)** of the sequence (vector) $\boldsymbol{c}$.
- Thus, the DFT of a vector $\boldsymbol{c}$ is just the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of the coordinates of $\boldsymbol{c}$ in the orthonormal basis $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$:

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.

- Let us now calculate the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ of a vector $\boldsymbol{c}$ in basis $\boldsymbol{\Phi}$.
- Since $\boldsymbol{\varphi}_m = \frac{1}{\sqrt{n}} \langle e^{i \frac{2\pi}{n} mk} : 0 \leq k < n \rangle$ using the definition of the scalar product in $\mathbb{C}^n$ we have

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} mk}$$

- Vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ is called the **Discrete Fourier Transform (DFT)** of the sequence (vector) $\boldsymbol{c}$.
- Thus, the DFT of a vector $\boldsymbol{c}$ is just the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of the coordinates of $\boldsymbol{c}$ in the orthonormal basis $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$:

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.

- Let us now calculate the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ of a vector $\boldsymbol{c}$ in basis $\boldsymbol{\Phi}$.
- Since $\boldsymbol{\varphi}_m = \frac{1}{\sqrt{n}} \langle e^{i \frac{2\pi}{n} m k} : 0 \leq k < n \rangle$ using the definition of the scalar product in $\mathbb{C}^n$ we have

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} m k}$$

- Vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ is called the **Discrete Fourier Transform (DFT)** of the sequence (vector) $\boldsymbol{c}$.
- Thus, the DFT of a vector $\boldsymbol{c}$ is just the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of the coordinates of $\boldsymbol{c}$ in the orthonormal basis $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$:

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.

- Let us now calculate the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ of a vector $\boldsymbol{c}$ in basis $\boldsymbol{\Phi}$.
- Since $\boldsymbol{\varphi}_m = \frac{1}{\sqrt{n}} \langle e^{i \frac{2\pi}{n} m k} : 0 \leq k < n \rangle$ using the definition of the scalar product in $\mathbb{C}^n$ we have

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} m k}$$

- Vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ is called the **Discrete Fourier Transform (DFT)** of the sequence (vector) $\boldsymbol{c}$.
- Thus, the DFT of a vector $\boldsymbol{c}$ is just the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of the coordinates of $\boldsymbol{c}$ in the orthonormal basis $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$:

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.

- Let us now calculate the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ of a vector $\boldsymbol{c}$ in basis $\boldsymbol{\Phi}$.
- Since $\boldsymbol{\varphi}_m = \frac{1}{\sqrt{n}} \langle e^{i \frac{2\pi}{n} m k} : 0 \leq k < n \rangle$ using the definition of the scalar product in $\mathbb{C}^n$ we have

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} m k}$$

- Vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ is called the **Discrete Fourier Transform (DFT)** of the sequence (vector) $\boldsymbol{c}$.
- Thus, the DFT of a vector $\boldsymbol{c}$ is just the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of the coordinates of $\boldsymbol{c}$ in the orthonormal basis $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$:

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

The DFT as a change of basis

- If we define vectors $\boldsymbol{\varphi}_k = \frac{\mathbf{f}_k}{\sqrt{n}}$, then these n vectors form an *orthonormal* basis, $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., a basis satisfying

$$\langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_m \rangle = \begin{cases} 0 & m \neq k \\ 1 & m = k \end{cases}$$

- In the remainder we want to explain why consider such a "strange" basis.

- Let us now calculate the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ of a vector $\boldsymbol{c}$ in basis $\boldsymbol{\Phi}$.
- Since $\boldsymbol{\varphi}_m = \frac{1}{\sqrt{n}} \langle e^{i \frac{2\pi}{n} m k} : 0 \leq k < n \rangle$ using the definition of the scalar product in $\mathbb{C}^n$ we have

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} m k}$$

- Vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of coordinates $\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle$ is called the **Discrete Fourier Transform (DFT)** of the sequence (vector) $\boldsymbol{c}$.
- Thus, the DFT of a vector $\boldsymbol{c}$ is just the vector $\widehat{\boldsymbol{c}} = (\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1})$ of the coordinates of $\boldsymbol{c}$ in the orthonormal basis $\boldsymbol{\Phi} = \{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$:

$$\boldsymbol{c} = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

The DFT as a change of basis

- Equating each coordinate of the lefthand side vector with the corresponding coordinate of the righthand side vector in

$$= \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

we obtain

$$c_k = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi k\,m}{n}} \tag{3}$$

- Note that the "forward" transform formula for calculating $\{\widehat{c}_m\}_{0\le m<n}$ from $\{c_k\}_{0\le k<n}$

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i\,\frac{2\pi}{n}\,m\,k}$$

and the inverse transform formula for calculating $\{c_k\}_{0\le k<n}$ from $\{\widehat{c}_m\}_{0\le m<n}$

$$c_k = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi k\,m}{n}}$$

differ only by the sign of the exponents of the complex exponentials $e^{\pm i\,\frac{2\pi k\,m}{n}}$

The DFT as a change of basis

- Equating each coordinate of the lefthand side vector with the corresponding coordinate of the righthand side vector in

$$= \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

we obtain

$$c_k = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi k\,m}{n}} \tag{3}$$

- Note that the "forward" transform formula for calculating $\{\widehat{c}_m\}_{0 \le m < n}$ from $\{c_k\}_{0 \le k < n}$

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i\,\frac{2\pi}{n}\,m\,k}$$

and the inverse transform formula for calculating $\{c_k\}_{0 \le k < n}$ from $\{\widehat{c}_m\}_{0 \le m < n}$

$$c_k = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi k\,m}{n}}$$

differ only by the sign of the exponents of the complex exponentials $e^{\pm i\,\frac{2\pi k\,m}{n}}$

The DFT as a change of basis

- Equating each coordinate of the lefthand side vector with the corresponding coordinate of the righthand side vector in

$$= \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m$$

we obtain

$$c_k = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi k\,m}{n}} \tag{3}$$

- Note that the "forward" transform formula for calculating $\{\widehat{c}_m\}_{0 \le m < n}$ from $\{c_k\}_{0 \le k < n}$

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i\,\frac{2\pi}{n}\,m\,k}$$

and the inverse transform formula for calculating $\{c_k\}_{0 \le k < n}$ from $\{\widehat{c}_m\}_{0 \le m < n}$

$$c_k = \sum_{m=0}^{n-1} \widehat{c}_m \boldsymbol{\varphi}_m(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi k\,m}{n}}$$

differ only by the sign of the exponents of the complex exponentials $e^{\pm i\,\frac{2\pi k\,m}{n}}$.

The DFT as a change of basis

- Also, by periodicity of the complex exponential e^{ix}, we have that for every **integer** k (but **NOT** for every real k!)

$$e^{-i\left(\frac{2\pi m}{n}k+2\pi k\right)} = e^{i\frac{2\pi(n-m)}{n}\cdot k}.$$

- Thus, for every integer k,

$$e^{i\frac{2\pi(n-m)}{n}\cdot k} = e^{-i\frac{2\pi m}{n}\cdot k} = \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$

- The last equality has two important consequences:
- The imaginary parts in the Fourier decomposition (1) will cancel out just in case $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$, because, then, for $t = k$, (k an integer), we get that for integers m, k, n, such that $1 \le m \le n-1$:

$$\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \widehat{c}_{n-m}\, e^{i\frac{2\pi(n-m)}{n}\cdot k} = \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m}\, \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$
$$= \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}}$$
$$= 2\mathcal{R}e(\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}).$$

- Thus, real discrete signals $\boldsymbol{c}$ of length n are precisely the signals which satisfy $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$ for all $1 \le k \le n-1$.

The DFT as a change of basis

- Also, by periodicity of the complex exponential e^{ix}, we have that for every **integer** k (but **NOT** for every real k!)

$$e^{-i\left(\frac{2\pi m}{n}k+2\pi k\right)} = e^{i\frac{2\pi(n-m)}{n}\cdot k}.$$

- Thus, for every integer

$$e^{i\frac{2\pi(n-m)}{n}\cdot k} = e^{-i\frac{2\pi m}{n}\cdot k} = \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$

- The last equality has two important consequences:
- The imaginary parts in the Fourier series (2) will cancel out just in case $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$, because, then, for $t = k$, (k an integer), we get that for integers m, k, n, such that $1 \leq m \leq n-1$:

$$\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \widehat{c}_{n-m}\, e^{i\frac{2\pi(n-m)}{n}\cdot k} = \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m}\, \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$
$$= \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}}$$
$$= 2\mathcal{R}e(\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}).$$

- Thus, real discrete signals $\boldsymbol{c}$ of length n are precisely the signals which satisfy $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$ for all $1 \leq k \leq n-1$.

The DFT as a change of basis

- Also, by periodicity of the complex exponential e^{ix}, we have that for every **integer** k (but **NOT** for every real k!)

$$e^{-i\left(\frac{2\pi m}{n}k+2\pi k\right)} = e^{i\frac{2\pi(n-m)}{n}\cdot k}.$$

- Thus, for every integer

$$e^{i\frac{2\pi(n-m)}{n}\cdot k} = e^{-i\frac{2\pi m}{n}\cdot k} = \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$

- The last equality has two important consequences:
- The imaginary parts in the Fourier decomposition (1) will cancel out just in case $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$, because, then, for $t = k$, (k an integer), we get that for integers m, k, n, such that $1 \le m \le n-1$:

$$\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \widehat{c}_{n-m}\, e^{i\frac{2\pi(n-m)}{n}\cdot k} = \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m}\, \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$
$$= \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}}$$
$$= 2\mathcal{R}e(\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}).$$

- Thus, real discrete signals $\boldsymbol{c}$ of length n are precisely the signals which satisfy $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$ for all $1 \le k \le n-1$.

The DFT as a change of basis

- Also, by periodicity of the complex exponential e^{ix}, we have that for every **integer** k (but **NOT** for every real k!)

$$e^{-i\left(\frac{2\pi m}{n}k+2\pi k\right)} = e^{i\frac{2\pi(n-m)}{n}\cdot k}.$$

- Thus, for every integer

$$e^{i\frac{2\pi(n-m)}{n}\cdot k} = e^{-i\frac{2\pi m}{n}\cdot k} = \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$

- The last equality has two important consequences:
- The imaginary parts in the right hand side sum in (3) will cancel out just in case $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$, because, then, for $t = k$, (k an integer), we get that for integers m, k, n, such that $n \geq m, k > 0$

$$\begin{aligned}\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \widehat{c}_{n-m}\, e^{i\frac{2\pi(n-m)}{n}\cdot k} &= \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m}\,\overline{e^{i\frac{2\pi m}{n}\cdot k}} \\ &= \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}} \\ &= 2\mathcal{R}e(\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}).\end{aligned}$$

- Thus, real discrete signals $\boldsymbol{c}$ of length n are precisely the signals which satisfy $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$ for all $1 \leq k \leq n-1$.

The DFT as a change of basis

- Also, by periodicity of the complex exponential e^{ix}, we have that for every **integer** k (but **NOT** for every real k!)

$$e^{-i\left(\frac{2\pi m}{n}k+2\pi k\right)} = e^{i\frac{2\pi(n-m)}{n}\cdot k}.$$

- Thus, for every integer k,

$$e^{i\frac{2\pi(n-m)}{n}\cdot k} = e^{-i\frac{2\pi m}{n}\cdot k} = \overline{e^{i\frac{2\pi m}{n}\cdot k}}$$

- The last equality has two important consequences:
- The imaginary parts in the right hand side sum in (3) will cancel out just in case $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$, because, then, for $t = k$, (k an integer), we get that for integers m, k, n, such that $n \geq m, k > 0$

$$\begin{aligned}\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \widehat{c}_{n-m}\, e^{i\frac{2\pi(n-m)}{n}\cdot k} &= \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m}\,\overline{e^{i\frac{2\pi m}{n}\cdot k}} \\ &= \widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k} + \overline{\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}} \\ &= 2\mathcal{R}e(\widehat{c}_m\, e^{i\frac{2\pi m}{n}\cdot k}).\end{aligned}$$

- Thus, real discrete signals $\boldsymbol{c}$ of length n are precisely the signals which satisfy $\widehat{c}_m = \overline{\widehat{c}_{n-m}}$ for all $1 \leq k \leq n-1$.

The DFT as a change of basis

- To answer the question why consider such a complicated basis $\mathbf{\Phi}$ we look at the complex sinusoids of the form:

$$\mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}k\cdot t} = \frac{1}{\sqrt{n}}\left(\cos\left(\frac{2\pi}{n}k\cdot t\right) + i\sin\left(\frac{2\pi}{n}k\cdot t\right)\right)$$

of frequencies $2\pi k/n$, for $k \leq n-1$.

- Then each of the basis vectors $\boldsymbol{\varphi}_k = 1/\sqrt{n}(\omega_n^{k\cdot 0}, \omega_n^{k\cdot 1}, \ldots, \omega_n^{k\cdot(n-1)})$ is just a sequence of samples of the function $\mathrm{Si}_k^n(t)$ evaluated at integers $t = 0, 1, \ldots, n-1$:

$$\mathrm{Si}_k^n = (\mathrm{Si}_k^n(0), \ldots, \mathrm{Si}_k^n(n-1))$$

- Thus, if we represent a vector $\boldsymbol{c}$ in such basis, i.e., as $\boldsymbol{c} = \sum_{k=0}^{n-1} \widehat{c}_k \boldsymbol{\varphi}_k$, we have represented $\boldsymbol{c}$ as a linear combination of samples of such complex sinusoids.
- If we also see the sequence $\boldsymbol{c}$ as a sequence of samples of a continuous time (i.e., analog) signal $c(t)$, then over the interval $[0, n-1]$

$$c(t) \approx \sum_{k=0}^{n-1} \widehat{c}_k \mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \widehat{c}_k e^{i\frac{2\pi k}{n}\cdot t} \tag{4}$$

where the equality is exact on integers $0, 1, \ldots, n-1$.

The DFT as a change of basis

- To answer the question why consider such a complicated basis $\mathbf{\Phi}$ we look at the complex sinusoids of the form:

$$\mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}k\cdot t} = \frac{1}{\sqrt{n}}\left(\cos\left(\frac{2\pi}{n}k\cdot t\right) + i\sin\left(\frac{2\pi}{n}k\cdot t\right)\right)$$

of frequencies $2\pi k/n$, for $0 \le k \le n-1$.

- Then each of the basis vectors $\boldsymbol{\varphi}_k = 1/\sqrt{n}(\omega_n^{k\cdot 0}, \omega_n^{k\cdot 1}, \ldots, \omega_n^{k\cdot(n-1)})$ is just a sequence of samples of the function $\mathrm{Si}_k^n(t)$, evaluated at integers $t = 0, 1, \ldots, n-1$:

$$\boldsymbol{\varphi}_k = (\mathrm{Si}_k^n(0), \ldots, \mathrm{Si}_k^n(n-1))$$

- Thus, if we represent a vector $\boldsymbol{c}$ in such a basis, i.e., as $\boldsymbol{c} = \sum_{k=0}^{n-1} \widehat{c}_k \boldsymbol{\varphi}_k$, we have represented $\boldsymbol{c}$ as a linear combination of samples of such complex sinusoids.
- If we also see the sequence $\boldsymbol{c}$ as a sequence of samples of a continuous time (i.e., analog) signal $c(t)$, then over the interval $[0, n-1]$

$$c(t) \approx \sum_{k=0}^{n-1} \widehat{c}_k \mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \widehat{c}_k e^{i\frac{2\pi k}{n}\cdot t} \tag{4}$$

where the equality is exact on integers $0, 1, \ldots, n-1$.

The DFT as a change of basis

- To answer the question why consider such a complex basis $\mathbf{\Phi}$ we look at the complex sinusoids of the form:

$$\mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}k\cdot t} = \frac{1}{\sqrt{n}}\left(\cos\left(\frac{2\pi}{n}k\cdot t\right) + i\sin\left(\frac{2\pi}{n}k\cdot t\right)\right)$$

of frequencies $2\pi k/n$, for $0 \le k \le n-1$.

- Then each of the basis vectors $\boldsymbol{\varphi}_k = 1/\sqrt{n}(\omega_n^{k\cdot 0}, \omega_n^{k\cdot 1}, \ldots, \omega_n^{k\cdot(n-1)})$ is just a sequence of samples of the function $\mathrm{Si}_k^n(t)$, evaluated at integers $t = 0, 1, \ldots, n-1$:

$$\boldsymbol{\varphi}_k = (\mathrm{Si}_k^n(0), \mathrm{Si}_k^n(1), \ldots, \mathrm{Si}_k^n(n-1))$$

- Thus, if we represent a vector $\boldsymbol{c}$ in such a basis, i.e., as $\boldsymbol{c} = \sum_{k=0}^{n-1} \widehat{c}_k\, \boldsymbol{\varphi}_k$, we have represented $\boldsymbol{c}$ as a linear combination of samples of such complex sinusoids.
- If we also see the sequence $\boldsymbol{c}$ as a sequence of samples of a continuous time (i.e., analog) signal $c(t)$, then over the interval $[0, n-1]$

$$c(t) \approx \sum_{k=0}^{n-1} \widehat{c}_k \mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \widehat{c}_k e^{i\frac{2\pi k}{n}\cdot t} \qquad (4)$$

where the equality is exact on integers $0, 1, \ldots, n-1$.

The DFT as a change of basis

- To answer the question why consider such a complex basis $\mathbf{\Phi}$ we look at the complex sinusoids of the form:

$$\mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}k\cdot t} = \frac{1}{\sqrt{n}}\left(\cos\left(\frac{2\pi}{n}k\cdot t\right) + i\sin\left(\frac{2\pi}{n}k\cdot t\right)\right)$$

of frequencies $2\pi k/n$, for $0 \le k \le n-1$.

- Then each of the basis vectors $\boldsymbol{\varphi}_k = 1/\sqrt{n}(\omega_n^{k\cdot 0}, \omega_n^{k\cdot 1}, \ldots, \omega_n^{k\cdot(n-1)})$ is just a sequence of samples of the function $\mathrm{Si}_k^n(t)$, evaluated at integers $t = 0, 1, \ldots, n-1$:

$$\boldsymbol{\varphi}_k = (\mathrm{Si}_k^n(0), \mathrm{Si}_k^n(1), \ldots, \mathrm{Si}_k^n(n-1))$$

- Thus, if we represent a vector $\boldsymbol{c}$ in such a basis, i.e., as $\boldsymbol{c} = \sum_{k=0}^{n-1} \widehat{c}_k\, \boldsymbol{\varphi}_k$, we have represented $\boldsymbol{c}$ as a linear combination of samples of such complex sinusoids.
- If we also see the sequence $\boldsymbol{c}$ as a sequence of samples of a continuous time (i.e., analog) signal $c(t)$, then over the interval $[0, n-1]$

$$c(t) \approx \sum_{k=0}^{n-1} \widehat{c}_k \mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \widehat{c}_k e^{i\frac{2\pi k}{n}\cdot t} \tag{4}$$

where the equality is exact on integers $0, 1, \ldots, n-1$.

The DFT as a change of basis

- Let us write each coordinate $\widehat{c}_k$ in the polar form: $\widehat{c}_k = |\widehat{c}_k| e^{i \arg(\widehat{c}_k)}$; then we get

$$c(t) \approx \sum_{k=0}^{n-1} |\widehat{c}_k| e^{i \arg} = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k)\right)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k)\right) + i \sin\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k)\right)\right).$$

- Note that the equality is exact only on integers $0, \ldots, n-1$, i.e., for m such that $0 \le m \le n-1$, we have

$$c(m) = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right) + i \sin\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right)\right)$$

The DFT as a change of basis

- Let us write each coordinate $\widehat{c}_k$ in the polar form: $\widehat{c}_k = |\widehat{c}_k| e^{i \arg(\widehat{c}_k)}$; then we get

$$c(t) \approx \sum_{k=0}^{n-1} |\widehat{c}_k| e^{i \arg} = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k)\right)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k)\right) + i \sin\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k)\right)\right).$$

- Note that the equality is exact only on integers $0, \ldots, n-1$, i.e., for m such that $0 \le m \le n-1$, we have

$$c(m) = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right) + i \sin\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right)\right)$$

The DFT as a change of basis

- Thus, we have obtained, called, a *spectral analysis* of $\boldsymbol{c}$, because in

$$c(m) = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right) + i \sin\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right) \right)$$

the values $|\widehat{c}(k)|/\sqrt{n}$ represent the *amplitudes* of the *harmonics*, i.e., complex sinusoids of frequencies $\frac{2\pi k}{n}$, and the arguments $\arg(\widehat{c}(k))$ represent the *phase shifts* of these complex sinusoids.

- The values of k, $0 \leq k < n$, are the indices of the corresponding *frequency bins*.

The DFT as a change of basis

- Thus, we have obtained, called, a *spectral analysis* of $\boldsymbol{c}$, because in

$$c(m) = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right) + i \sin\left(\frac{2\pi k}{n} \cdot m + \arg(\widehat{c}_k)\right) \right)$$

the values $|\widehat{c}(k)|/\sqrt{n}$ represent the *amplitudes* of the *harmonics*, i.e., complex sinusoids of frequencies $\frac{2\pi k}{n}$, and the arguments $\arg(\widehat{c}(k))$ represent the *phase shifts* of these complex sinusoids.

- The values of k, $0 \leq k < n$, are the indices of the corresponding *frequency bins*.

The DFT as a change of basis

- Approximation

$$c(t) = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i(\quad)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k) \right) + i \sin\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k) \right) \right)$$

has two important shortcomings:

1. Evan if $\boldsymbol{c}$ is real, $c(t)$ attains complex values for non integer values of t.
2. It unnecessarily involves complex exponentials of frequencies larger than π.

The DFT as a change of basis

- Approximation

$$c(t) = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i(\quad)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k) \right) + i \sin\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k) \right) \right)$$

has two important shortcomings:

1. Evan if $\boldsymbol{c}$ is real, $c(t)$ attains complex values for non integer values of t.
2. It unnecessarily involves complex exponentials of frequencies larger than π.

The DFT as a change of basis

- Approximation

$$c(t) = \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} e^{i(\quad)}$$

$$= \sum_{k=0}^{n-1} \frac{|\widehat{c}_k|}{\sqrt{n}} \left(\cos\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k) \right) + i \sin\left(\frac{2\pi k}{n} \cdot t + \arg(\widehat{c}_k) \right) \right)$$

has two important shortcomings:

1. Evan if $\boldsymbol{c}$ is real, $c(t)$ attains complex values for non integer values of t.
2. It unnecessarily involves complex exponentials of frequencies larger than π.

The DFT as a change of basis

- This can easily be remedied by allowing complex exponentials of negative frequencies, thus letting

$$\mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}k\cdot t} = \frac{1}{\sqrt{n}}\left(\cos\left(\frac{2\pi}{n}k\cdot t\right) + i\sin\left(\frac{2\pi}{n}k\cdot t\right)\right)$$

for all k such that $-\lfloor (n-1)/2 \rfloor \le k \le \lfloor (n-1)/2 \rfloor$, and, if n is even, also

$$\mathrm{Si}_{n/2}^n(t) = \frac{1}{\sqrt{n}}\cos(\pi t)\,.$$

- These basis functions have frequencies at most π.
- For real valued $\boldsymbol{c}$, they result in a real valued interpolation function which is exact on the integers $0 \ldots n-1$:
  - for even n:

$$c(t) \approx \frac{1}{\sqrt{n}}\Big(\widehat{c}(0) + \widehat{c}(1)e^{i\frac{2\pi}{n}t} + \ldots + \widehat{c}(n/2-1)e^{i\frac{2\pi}{n}(n/2-1)t} + \widehat{c}(n/2)\cos\pi t + \\ + \widehat{c}(n/2+1)e^{-i\frac{2\pi}{n}(n/2-1)t} + \ldots + \widehat{c}(n-1)e^{-i\frac{2\pi}{n}t}\Big)$$

The DFT as a change of basis

- This can easily be remedied by allowing complex exponentials of negative frequencies, thus letting

$$\mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}k\cdot t} = \frac{1}{\sqrt{n}}\left(\cos\left(\frac{2\pi}{n}k\cdot t\right) + i\sin\left(\frac{2\pi}{n}k\cdot t\right)\right)$$

for all k such that $-\lfloor (n-1)/2 \rfloor \le k \le \lfloor (n-1)/2 \rfloor$, and, if n is even, also

$$\mathrm{Si}_{n/2}^n(t) = \frac{1}{\sqrt{n}}\cos(\pi t)\,.$$

- These basis functions have frequencies at most π.
- For real valued $\boldsymbol{c}$, they result in a real valued interpolation function which is exact on the integers $0 \ldots n-1$:
  - for even n:

$$c(t) \approx \frac{1}{\sqrt{n}}\Big(\widehat{c}(0) + \widehat{c}(1)e^{i\frac{2\pi}{n}t} + \ldots + \widehat{c}(n/2-1)e^{i\frac{2\pi}{n}(n/2-1)t} + \widehat{c}(n/2)\cos\pi t + \\ + \widehat{c}(n/2+1)e^{-i\frac{2\pi}{n}(n/2-1)t} + \ldots + \widehat{c}(n-1)e^{-i\frac{2\pi}{n}t}\Big)$$

The DFT as a change of basis

- This can easily be remedied by allowing complex exponentials of negative frequencies, thus letting

$$\mathrm{Si}_k^n(t) = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}k\cdot t} = \frac{1}{\sqrt{n}}\left(\cos\left(\frac{2\pi}{n}k\cdot t\right) + i\sin\left(\frac{2\pi}{n}k\cdot t\right)\right)$$

for all k such that $-\lfloor (n-1)/2 \rfloor \le k \le \lfloor (n-1)/2 \rfloor$, and, if n is even, also

$$\mathrm{Si}_{n/2}^n(t) = \frac{1}{\sqrt{n}}\cos(\pi t)\,.$$

- These basis functions have frequencies at most π.
- For real valued $\boldsymbol{c}$, they result in a real valued interpolation function which is exact on the integers $0 \ldots n-1$:
  - for even n:

$$c(t) \approx \frac{1}{\sqrt{n}}\Big(\widehat{c}(0) + \widehat{c}(1)e^{i\frac{2\pi}{n}t} + \ldots + \widehat{c}(n/2-1)e^{i\frac{2\pi}{n}(n/2-1)t} + \widehat{c}(n/2)\cos\pi t + \\ + \widehat{c}(n/2+1)e^{-i\frac{2\pi}{n}(n/2-1)t} + \ldots + \widehat{c}(n-1)e^{-i\frac{2\pi}{n}t}\Big)$$

The DFT as a change of basis

– for odd n:

$$c(t) \approx \frac{1}{\sqrt{n}} \left(\widehat{c}(0) + \widehat{c}(1) e^{i \frac{2\pi}{n} t} + \ldots + \widehat{c}(\lfloor n/2 \rfloor) e^{i \frac{2\pi}{n} \lfloor n/2 \rfloor t} + \right.$$

$$\left. \widehat{c}(\lfloor n/2 \rfloor + 1) e^{-i \frac{2\pi}{n} \lfloor n/2 \rfloor t} + \ldots + \widehat{c}(n-1) e^{-i \frac{2\pi}{n} t} \right)$$

- Note that the formula for the DFT

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} m k} = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k \left(e^{-i \frac{2\pi}{n} m} \right)^k$$

can be seen as an evaluation of polynomial

$$p(x) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k x^k$$

at

$$x_m = e^{-i \frac{2\pi}{n} m} = \omega_n^{-m}$$

The DFT as a change of basis

– for odd n:

$$c(t) \approx \frac{1}{\sqrt{n}} \left(\widehat{c}(0) + \widehat{c}(1) e^{i \frac{2\pi}{n} t} + \ldots + \widehat{c}(\lfloor n/2 \rfloor) e^{i \frac{2\pi}{n} \lfloor n/2 \rfloor t} + \right.$$

$$\left. \widehat{c}(\lfloor n/2 \rfloor + 1) e^{-i \frac{2\pi}{n} \lfloor n/2 \rfloor t} + \ldots + \widehat{c}(n-1) e^{-i \frac{2\pi}{n} t} \right)$$

- Note that the formula for the DFT

$$\widehat{c}_m = \langle \boldsymbol{c}, \boldsymbol{\varphi}_m \rangle = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i \frac{2\pi}{n} m k} = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k \left(e^{-i \frac{2\pi}{n} m} \right)^k$$

can be seen as an evaluation of polynomial

$$C(x) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k x^k$$

at

$$x_m = e^{-i \frac{2\pi}{n} m} = \omega_n^{-m}$$

The DFT as a change of basis

- Thus, with $C(x) = \frac{1}{\sqrt{n}}$, 

$$\langle \widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1} \rangle = \langle C(\omega_n^0), C(\omega_n^{-1}), \ldots, C(\omega_n^{-(n-1)}) \rangle$$

- One of the main features of the DFT is that it can be computed very efficiently using the Fast Fourier Transform algorithm (FFT).
- Such algorithm is based on the fact that $\omega_{2n}^{2k} = \omega_n^k$.
- Thus, as k ranges from 0 to $2n-1$ the squares of ω_{2n}^{-k} range through ω_n^{-k} and there are only n distinct such values.

The DFT as a change of basis

- Thus, with $C(x) = \frac{1}{\sqrt{n}}$  ,

$$\langle \widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1} \rangle = \langle C(\omega_n^0), C(\omega_n^{-1}), \ldots, C(\omega_n^{-(n-1)}) \rangle$$

- One of the main features of the DFT is that it can be evaluated very efficiently using the Fast Fourier Transform algorithm (FFT).
- Such algorithm is based on the fact that $(\omega_{2n}^{-k})^2 = \omega_{2n}^{-2k} = \omega_n^{-k}$.
- Thus, as k ranges from 0 to $2n-1$ the squares of ω_{2n}^{-k} range through ω_n^{-k} and there are only n distinct such values.

The DFT as a change of basis

- Thus, with $C(x) = \frac{1}{\sqrt{n}}$ [redacted],

$$\langle \widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1} \rangle = \langle C(\omega_n^0), C(\omega_n^{-1}), \ldots, C(\omega_n^{-(n-1)}) \rangle$$

- One of the main features of the DFT is that it can be evaluated very efficiently using the Fast Fourier Transform algorithm (FFT).
- Such algorithm is based on the fact that $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = \omega_n^k$.
- Thus, as k ranges from 0 to $2n-1$ the squares of ω_{2n}^{-k} range through ω_n^{-k} and there are only n distinct such values.

The DFT as a change of basis

- Thus, with $C(x) = \frac{1}{\sqrt{n}}$ [redacted],

$$\langle \widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_{n-1} \rangle = \langle C(\omega_n^0), C(\omega_n^{-1}), \ldots, C(\omega_n^{-(n-1)}) \rangle$$

- One of the main features of the DFT is that it can be evaluated very efficiently using the Fast Fourier Transform algorithm (FFT).

- Such algorithm is based on the fact that $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = \omega_n^k$.

- Thus, as k ranges from 0 to $2n-1$ the squares of ω_{2n}^{-k} range through ω_n^{-k} and there are only n distinct such values.

The Fast Fourier Transform (FFT)

- **The main idea of the FFT algorithm** is divide-and-conquer by splitting the polynomial into the even powers and the odd powers:

$$\begin{aligned} C(x) &= (c_0 + c_2 x^2 + \ldots + c_{n-2} x^{n-2}) + (c_1 x + c_3 x^3 + \ldots c_{n-1} x^{n-1}) \\ &= c_0 + c_2 x^2 + \ldots + c_{n-2}(x^2)^{\frac{n}{2}-1} \\ &\quad + x\left(c_1 + c_3 x^2 + c_5 (x^2)^2 + \ldots + c_{n-1}(x^2)^{\frac{n}{2}-1}\right) \end{aligned}$$

- Let us define

$$C^{[0]}(y) = c_0 + c_2 y + c_4 y^2 + \ldots + c_{n-2} y^{\frac{n}{2}-1}$$
$$C^{[1]}(y) = c_1 + c_3 y + c_5 y^2 + \ldots + c_{n-1} y^{\frac{n}{2}-1}$$

- Then

$$C(x) = C^{[0]}(x^2) + x\, C^{[1]}(x^2)$$

- Note that the number of coefficients of the polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is $n/2$ each, while the number of coefficients of the polynomial $C(x)$ is n. Thus, the number of coefficients of each of the two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is only one half of the number of coefficients of the polynomial $C(x)$.

The Fast Fourier Transform (FFT)

- **The main idea of the FFT algorithm** is divide-and-conquer by splitting the polynomial into the even powers and the odd powers:

$$\begin{aligned} C(x) &= (c_0 + c_2x^2 + \ldots + c_{n-2}x^{n-2}) + (c_1x + c_3x^3 + \ldots c_{n-1}x^{n-1}) \\ &= c_0 + c_2x^2 + \ldots + c_{n-2}(x^2)^{\frac{n}{2}-1} \\ &\quad + x\left(c_1 + c_3x^2 + c_5(x^2)^2 + \ldots + c_{n-1}(x^2)^{\frac{n}{2}-1}\right) \end{aligned}$$

- Let us define

$$C^{[0]}(y) = c_0 + c_2y + c_4y^2 + \ldots + c_{n-2}y^{\frac{n}{2}-1}$$
$$C^{[1]}(y) = c_1 + c_3y + c_5y^2 + \ldots + c_{n-1}y^{\frac{n}{2}-1}$$

- Then

$$C(x) = C^{[0]}(x^2) + xC^{[1]}(x^2)$$

- Note that the number of coefficients of the polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is $n/2$ each, while the number of coefficients of the polynomial $C(x)$ is n. Thus, the number of coefficients of each of the two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is only one half of the number of coefficients of the polynomial $C(x)$.

The Fast Fourier Transform (FFT)

- **The main idea of the FFT algorithm** is a divide-and-conquer by splitting the polynomial into the even powers and the odd powers:

$$\begin{aligned} C(x) &= (c_0 + c_2 x^2 + \ldots + c_{n-2} x^{n-2}) + (c_1 x + c_3 x^3 + \ldots c_{n-1} x^{n-1}) \\ &= c_0 + c_2 x^2 + \ldots + c_{n-2}(x^2)^{\frac{n}{2}-1} \\ &\quad + x\left(c_1 + c_3 x^2 + c_5 (x^2)^2 + \ldots + c_{n-1}(x^2)^{\frac{n}{2}-1}\right) \end{aligned}$$

- Let us define

$$C^{[0]}(y) = c_0 + c_2 y + c_4 y^2 + \ldots + c_{n-2} y^{\frac{n}{2}-1}$$
$$C^{[1]}(y) = c_1 + c_3 y + c_5 y^2 + \ldots + c_{n-1} y^{\frac{n}{2}-1}$$

- Then

$$C(x) = C^{[0]}(x^2) + x\, C^{[1]}(x^2)$$

- Note that the number of coefficients of the polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is $n/2$ each, while the number of coefficients of the polynomial $C(x)$ is n. Thus, the number of coefficients of each of the two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is only one half of the number of coefficients of the polynomial $C(x)$.

The Fast Fourier Transform (FFT)

- **The main idea of the FFT algorithm** is divide-and-conquer by splitting the polynomial into the even powers and the odd powers:

$$\begin{aligned} C(x) &= (c_0 + c_2x^2 + \ldots + c_{n-2}x^{n-2}) + (c_1x + c_3x^3 + \ldots c_{n-1}x^{n-1}) \\ &= c_0 + c_2x^2 + \ldots + c_{n-2}(x^2)^{\frac{n}{2}-1} \\ &\quad + x\left(c_1 + c_3x^2 + c_5(x^2)^2 + \ldots + c_{n-1}(x^2)^{\frac{n}{2}-1}\right) \end{aligned}$$

- Let us define

$$C^{[0]}(y) = c_0 + c_2y + c_4y^2 + \ldots + c_{n-2}y^{\frac{n}{2}-1}$$
$$C^{[1]}(y) = c_1 + c_3y + c_5y^2 + \ldots + c_{n-1}y^{\frac{n}{2}-1}$$

- Then

$$C(x) = C^{[0]}(x^2) + x\,C^{[1]}(x^2)$$

- Note that the number of coefficients of the polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is $n/2$ each, while the number of coefficients of the polynomial $C(x)$ is n. Thus, the number of coefficients of each of the two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ is only one half of the number of coefficients of the polynomial $C(x)$.

The Fast Fourier Transform (FFT)

- **Problem of size n:**
  Evaluate a polynomial with n coefficients at n many complex conjugates of the roots of unity of order n.

- **Problem of size $n/2$:**
  Evaluate a polynomial with $n/2$ coefficients at $n/2$ complex conjugates of the roots of unity of order $n/2$.

- We reduced evaluation of one polynomial $A(x)$ with n coefficients at inputs $x = \omega_n^0,\ x = \omega_n^{-1},\ x = \omega_n^{-2}, \ldots, x = \omega_n^{-(n-1)}$ to evaluation of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ each with $n/2$ coefficients, at points $y = x^2$ for the same values of inputs x.

- However, as x ranges through values $\{\omega_n^0, \omega_n^{-1}, \omega_n^{-2}, \ldots, \omega_n^{-(n-1)}\}$, the value of $y = x^2$ ranges through $\{\omega_{\frac{n}{2}}^0, \omega_{\frac{n}{2}}^{-1}, \omega_{\frac{n}{2}}^{-2}, \ldots, \omega_{\frac{n}{2}}^{-(n-1)}\}$, and **there are only $n/2$ distinct such values.**

The Fast Fourier Transform (FFT)

- **Problem of size n:**
  Evaluate a polynomial with n coefficients at n many complex conjugates of the roots of unity of order n.
- **Problem of size $n/2$:**
  Evaluate a polynomial with $n/2$ coefficients at $n/2$ complex conjugates of the roots of unity of order $n/2$.

- We reduced evaluation of one polynomial $A(x)$ with n coefficients at inputs $x = \omega_n^0,\ x = \omega_n^{-1},\ x = \omega_n^{-2}, \ldots, x = \omega_n^{-(n-1)}$ to evaluation of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ each with $n/2$ coefficients, at points $y = x^2$ for the same values of inputs x.
- However, as x ranges through values $\{\omega_n^0, \omega_n^{-1}, \omega_n^{-2}, \ldots, \omega_n^{-(n-1)}\}$, the value of $y = x^2$ ranges through $\{\omega_{\frac{n}{2}}^0, \omega_{\frac{n}{2}}^{-1}, \omega_{\frac{n}{2}}^{-2}, \ldots, \omega_{\frac{n}{2}}^{-(n-1)}\}$, and **there are only $n/2$ distinct such values.**

The Fast Fourier Transform (FFT)

- **Problem of size n:**
  Evaluate a polynomial with n coefficients at n many complex conjugates of the roots of unity of order n.
- **Problem of size $n/2$:**
  Evaluate a polynomial with $n/2$ coefficients at $n/2$ complex conjugates of the roots of unity of order $n/2$.
- We reduced evaluation of our polynomial $C(x)$ with n coefficients at inputs $x = \omega_n^0,\ x = \omega_n^{-1},\ x = \omega_n^{-2}, \ldots, x = \omega_n^{-(n-1)}$ to evaluation of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ each with $n/2$ coefficients, at points $y = x^2$ for the same values of inputs x.
- However, as x ranges through values $\{\omega_n^0, \omega_n^{-1}, \omega_n^{-2}, \ldots, \omega_n^{-(n-1)}\}$, the value of $y = x^2$ ranges through $\{\omega_{\frac{n}{2}}^0, \omega_{\frac{n}{2}}^{-1}, \omega_{\frac{n}{2}}^{-2}, \ldots, \omega_{\frac{n}{2}}^{-(n-1)}\}$, and **there are only $n/2$ distinct such values.**

The Fast Fourier Transform (FFT)

- **Problem of size n:**
  Evaluate a polynomial with n coefficients at n many complex conjugates of the roots of unity of order n.
- **Problem of size $n/2$:**
  Evaluate a polynomial with $n/2$ coefficients at $n/2$ complex conjugates of the roots of unity of order $n/2$.

- We reduced evaluation of our polynomial $C(x)$ with n coefficients at inputs $x = \omega_n^0,\ x = \omega_n^{-1},\ x = \omega_n^{-2}, \ldots, x = \omega_n^{-(n-1)}$ to evaluation of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$ each with $n/2$ coefficients, at points $y = x^2$ for the same values of inputs x.
- However, as x ranges through values $\{\omega_n^0, \omega_n^{-1}, \omega_n^{-2}, \ldots, \omega_n^{-(n-1)}\}$, the value of $y = x^2$ ranges through $\{\omega_{\frac{n}{2}}^0, \omega_{\frac{n}{2}}^{-1}, \omega_{\frac{n}{2}}^{-2}, \ldots, \omega_{\frac{n}{2}}^{-(n-1)}\}$, and **there are only $n/2$ distinct such values**.

The Fast Fourier Transform (FFT)

- Once we get these $n/2$ $C^{[0]}(x^2)$ and $C^{[1]}(x^2)$ we need n additional multiplications with nu to obtain the values of

$$C(\omega_n^{-k}) = C^{[0]}((\omega_n^{-k})^2) + \omega_n^{-k} \cdot C^{[1]}((\omega_n^{-k})^2)$$
$$= C^{[0]}(\omega_{n/2}^{-k}) + \omega_n^{-k} \cdot C^{[1]}(\omega_{n/2}^{-k}).$$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- We can assume that the length n of the sequence c is of the form $n = 2^k$ because we can always pad it with less than n zeros to make it of length which is a power of 2.

The Fast Fourier Transform (FFT)

- Once we get these $n/2$ $C^{[0]}(x^2)$ and $C^{[1]}(x^2)$ we need n additional multiplications with n to obtain the values of

$$C(\omega_n^{-k}) = C^{[0]}((\omega_n^{-k})^2) + \omega_n^{-k} \cdot C^{[1]}((\omega_n^{-k})^2)$$
$$= C^{[0]}(\omega_{n/2}^{-k}) + \omega_n^{-k} \cdot C^{[1]}(\omega_{n/2}^{-k}).$$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.

- We can assume that the length n of the sequence c is of the form $n = 2^k$ because we can always pad it with less than n zeros to make it of length which is a power of 2.

The Fast Fourier Transform (FFT)

- Once we get these $n/2$ $C^{[0]}(x^2)$ and $C^{[1]}(x^2)$ we need n additional multiplications with ω_n^{-k} to obtain the values of

$$C(\omega_n^{-k}) = C^{[0]}((\omega_n^{-k})^2) + \omega_n^{-k} \cdot C^{[1]}((\omega_n^{-k})^2)$$
$$= C^{[0]}(\omega_{n/2}^{-k}) + \omega_n^{-k} \cdot C^{[1]}(\omega_{n/2}^{-k}).$$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- We can assume that the length n of the sequence c is of the form $n = 2^k$ because we can always pad it with fewer than n zeros to make it of length which is a power of 2.

The Fast Fourier Transform (FFT) - a simplification

- Note that $\omega_n^{-\frac{n}{2}} = \omega_{2\frac{n}{2}}^{-\frac{n}{2}} = \omega_2^{-1} = (-1)^{-1} = -1$, thus,

$$\omega_n^{-(k+\frac{n}{2})} = \omega_n^{-\frac{n}{2}} \omega_n^{-k} = \omega_2 \omega_n^{-k} = -\omega_n^{-k};$$

- We can now simplify evaluation for $n/2 \le k < n$ as follows: let $k = \frac{n}{2} + m$ where $0 \le m < n/2$; then

$$C(\omega_n^{-k}) = C^{[0]}((\omega_n^{-k})^2) + \omega_n^{-k} C^{[1]}((\omega_n^{-k})^2)$$

$$C\left(\omega_n^{-(\frac{n}{2}+m)}\right) = C^{[0]}\left(\left(\omega_n^{-(\frac{n}{2}+m)}\right)^2\right) + \omega_n^{-(\frac{n}{2}+m)} C^{[1]}\left(\left(\omega_n^{-(\frac{n}{2}+m)}\right)^2\right)$$

$$= C^{[0]}\left(\omega_{n/2}^{-(\frac{n}{2}+m)}\right) + \omega_n^{-\frac{n}{2}} \omega_n^{-m} C^{[1]}\left(\omega_{n/2}^{-(\frac{n}{2}+m)}\right)$$

$$= C^{[0]}\left(\omega_{n/2}^{-n/2}\omega_{n/2}^{-m}\right) + \omega_n^{-\frac{n}{2}} \omega_n^{-m} C^{[1]}\left(\omega_{n/2}^{-n/2}\omega_{n/2}^{-m}\right)$$

$$= C^{[0]}\left(\omega_{n/2}^{-m}\right) + \omega_2\, \omega_n^{-m} C^{[1]}\left(\omega_{n/2}^{-m}\right)$$

$$= C^{[0]}\left(\omega_{n/2}^{-m}\right) - \omega_n^{-m} C^{[1]}\left(\omega_{n/2}^{-m}\right)$$

The Fast Fourier Transform (FFT) - a simplification

- Note that $\omega_n^{-\frac{n}{2}} = \omega_{2\frac{n}{2}}^{-\frac{n}{2}} = \omega_2^{-1} = (-1)^{-1} = -1$; thus,

$$\omega_n^{-(\frac{n}{2}+k)} = \omega_n^{-\frac{n}{2}}\omega_n^{-k} = \omega_2\omega_n^{-k} = -\omega_n^{-k};$$

- We can now simplify eva

$$C(\omega_n^{-k}) = C^{[0]}((\omega_n^{-k})^2) + \omega_n^{-k}C^{[1]}((\omega_n^{-k})^2)$$

for $n/2 \le k < n$ as follows: let $k = \frac{n}{2} + m$ where $0 \le m < n/2$; then

$$\begin{aligned}
C\left(\omega_n^{-(\frac{n}{2}+m)}\right) &= C^{[0]}\left(\left(\omega_n^{-(\frac{n}{2}+m)}\right)^2\right) + \omega_n^{-(\frac{n}{2}+m)}C^{[1]}\left(\left(\omega_n^{-(\frac{n}{2}+m)}\right)^2\right) \\
&= C^{[0]}\left(\omega_{n/2}^{-(\frac{n}{2}+m)}\right) + \omega_n^{-\frac{n}{2}}\omega_n^{-m}C^{[1]}\left(\omega_{n/2}^{-(\frac{n}{2}+m)}\right) \\
&= C^{[0]}\left(\omega_{n/2}^{-n/2}\omega_{n/2}^{-m}\right) + \omega_{2\frac{n}{2}}^{-\frac{n}{2}}\omega_n^{-m}C^{[1]}\left(\omega_{n/2}^{-n/2}\omega_{n/2}^{-m}\right) \\
&= C^{[0]}\left(\omega_{n/2}^{-m}\right) + \omega_2\,\omega_n^{-m}C^{[1]}\left(\omega_{n/2}^{-m}\right) \\
&= C^{[0]}\left(\omega_{n/2}^{-m}\right) - \omega_n^{-m}C^{[1]}\left(\omega_{n/2}^{-m}\right)
\end{aligned}$$

- So we have for $0 \le m <$

$$C^{[0]}(\omega_{n/2}^{-m}) + \omega_n^{-m} C^{[1]}(\omega_{n/2}^{-m})$$

$$C\left(\omega_n^{-\left(\frac{n}{2}+m\right)}\right) = C^{[0]}\left(\omega_{n/2}^{-m}\right) - \omega_n^{-m} C^{[1]}\left(\omega_{n/2}^{-m}\right)$$

- These two sets of evalua... $k = 0$ to $k = n/2 - 1$ can replace evaluations for $k = 0$ to $k = n - 1$ of

$$C(\omega_n^k) = C^{[0]}(\omega_{n/2}^k) + \omega_n^k C^{[1]}(\omega_{n/2}^k)$$

- We can now write a pseu... FFT algorithm:

The Fast Fourier Transform (FFT) - a simplification

- So we have for $0 \leq m <$ 

$$C^{[0]}(\omega_{n/2}^{-m}) + \omega_n^{-m} C^{[1]}(\omega_{n/2}^{-m})$$

$$C\left(\omega_n^{-\left(\frac{n}{2}+m\right)}\right) = C^{[0]}\left(\omega_{n/2}^{-m}\right) - \omega_n^{-m} C^{[1]}\left(\omega_{n/2}^{-m}\right)$$

- These two sets of evaluations for $k = 0$ to $k = n/2 - 1$ can replace evaluations for $k = 0$ to $k = n - 1$ of

$$C(\omega_n^k) = C^{[0]}(\omega_{n/2}^k) + \omega_n^k C^{[1]}(\omega_{n/2}^k)$$

- We can now write a pseudo-code for the FFT algorithm:

The Fast Fourier Transform (FFT) - a simplification

- So we have for $0 \leq m <$

$$C^{[0]}(\omega_{n/2}^{-m}) + \omega_n^{-m} C^{[1]}(\omega_{n/2}^{-m})$$

$$C\left(\omega_n^{-\left(\frac{n}{2}+m\right)}\right) = C^{[0]}\left(\omega_{n/2}^{-m}\right) - \omega_n^{-m} C^{[1]}\left(\omega_{n/2}^{-m}\right)$$

- These two sets of evaluations for $k = 0$ to $k = n/2 - 1$ can replace evaluations for $k = 0$ to $k = n - 1$ of

$$C(\omega_n^k) = C^{[0]}(\omega_{n/2}^k) + \omega_n^k C^{[1]}(\omega_{n/2}^k)$$

- We can now write a pseudo-code for our FFT algorithm:

FFT algorithm

```
1: function FFT*(C)
2:     n ← length[C]
3:     if n = 1 then return C
4:     else
5:         C^[0] ← (c_0, c_2, ... c_{r-1})
6:         C^[1] ← (c_1, c_3, ... c_{r-1})
7:         y^[0] ← FFT(C^[0]);
8:         y^[1] ← FFT(C^[1]);
9:         ω_n ← e^{i 2π/n};
10:        ω ← 1;
11:        for k = 0 to k = n/2 − 1 do:
12:            y_k ← y_k^[0] + ω · y_k^[1];
13:            y_{n/2+k} ← y_k^[0] − ω · y_k^[1];
14:            ω ← ω · ω_n;
15:        end for
16:    end if
17:    return y
18: end function
```

```
1: function FFT(C)
2:     return FFT*(C)/√length(C);
3: end function
```

How fast is the Fast Fourier Transform?

- We have recursively reduced evaluation of a polynomial $C(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$, each with $n/2$ coefficients, at $n/2$ many roots of unity of order $n/2$.
- Once we get these $n/2$ values for $C^{[0]}(y)$ and $C^{[1]}(y)$ we need $n/2$ additional multiplications to obtain the values of

$$\underbrace{C(\omega_n^k)}_{y_k} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} + \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{5}$$

and

$$\underbrace{C(\omega_n^{\frac{n}{2}+k})}_{y_{\frac{n}{2}+k}} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} - \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{6}$$

for all $0 \le k < n/2$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2\,T\left(\frac{n}{2}\right) + c\,n$$

- The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

- We have recursively reduced evaluation of a polynomial $C(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$, each with $n/2$ coefficients, at $n/2$ many roots of unity of order $n/2$.
- Once we get these $n/2$ values for $C^{[0]}(y)$ and $C^{[1]}(y)$ we need $n/2$ additional multiplications to obtain values of

$$\underbrace{C(\omega_n^k)}_{y_k} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} + \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{5}$$

and

$$\underbrace{C(\omega_n^{\frac{n}{2}+k})}_{y_{\frac{n}{2}+k}} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} - \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{6}$$

for all $0 \le k < n/2$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2\,T\left(\frac{n}{2}\right) + c\,n$$

- The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

- We have recursively reduced evaluation of a polynomial $C(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$, each with $n/2$ coefficients, at $n/2$ many roots of unity of order $n/2$.
- Once we get these $n/2$ values for $C^{[0]}(y)$ and $C^{[1]}(y)$ we need $n/2$ additional multiplications to obtain values of

$$\underbrace{C(\omega_n^k)}_{y_k} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} + \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{5}$$

and

$$\underbrace{C(\omega_n^{\frac{n}{2}+k})}_{y_{\frac{n}{2}+k}} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} - \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{6}$$

for all $0 \leq k < n/2$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2\,T\left(\frac{n}{2}\right) + c\,n$$

- The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

- We have recursively reduced evaluation of a polynomial $C(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$, each with $n/2$ coefficients, at $n/2$ many roots of unity of order $n/2$.
- Once we get these $n/2$ values for $C^{[0]}(y)$ and $C^{[1]}(y)$ we need $n/2$ additional multiplications to obtain values of

$$\underbrace{C(\omega_n^k)}_{y_k} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} + \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{5}$$

and

$$\underbrace{C(\omega_n^{\frac{n}{2}+k})}_{y_{\frac{n}{2}+k}} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} - \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{6}$$

for all $0 \leq k < n/2$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2\,T\left(\frac{n}{2}\right) + c\,n$$

- The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

- We have recursively reduced evaluation of a polynomial $C(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$, each with $n/2$ coefficients, at $n/2$ many roots of unity of order $n/2$.
- Once we get these $n/2$ values for $C^{[0]}(y)$ and $C^{[1]}(y)$ we need $n/2$ additional multiplications to obtain values of

$$\underbrace{C(\omega_n^k)}_{y_k} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} + \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{5}$$

and

$$\underbrace{C(\omega_n^{\frac{n}{2}+k})}_{y_{\frac{n}{2}+k}} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} - \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{6}$$

for all $0 \leq k < n/2$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2\,T\left(\frac{n}{2}\right) + c\,n$$

- The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

- We have recursively reduced evaluation of a polynomial $C(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $C^{[0]}(y)$ and $C^{[1]}(y)$, each with $n/2$ coefficients, at $n/2$ many roots of unity of order $n/2$.
- Once we get these $n/2$ values for $C^{[0]}(y)$ and $C^{[1]}(y)$ we need $n/2$ additional multiplications to obtain values of

$$\underbrace{C(\omega_n^k)}_{y_k} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} + \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{5}$$

and

$$\underbrace{C(\omega_n^{\frac{n}{2}+k})}_{y_{\frac{n}{2}+k}} = \underbrace{C^{[0]}(\omega_{n/2}^k)}_{y_k^{[0]}} - \omega_n^k \underbrace{C^{[1]}(\omega_{n/2}^k)}_{y_k^{[1]}} \tag{6}$$

for all $0 \leq k < n/2$

- Thus, we have reduced a problem of size n to two such problems of size $n/2$, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2\,T\left(\frac{n}{2}\right) + c\,n$$

- The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

- Recall that we had

$$\widehat{c}_m = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i\,\frac{2\pi}{n}\,m\,k}$$

$$c_k = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi}{n}\,m\,k}$$

- Thus, the same algorithm can be used to find the DFT of a sequence as well as the IDFT (Inverse Discrete Fourier Transform) of a sequence, with the only change of replacing ω_n with ω_n^{-1}.

How fast is the Fast Fourier Transform?

- Recall that we had

$$\widehat{c}_m = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} c_k e^{-i\,\frac{2\pi}{n}\,m\,k}$$

$$c_k = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} \widehat{c}_m e^{i\,\frac{2\pi}{n}\,m\,k}$$

- Thus, the same algorithm can be used to find the DFT of a sequence as well as the IDFT (Inverse Discrete Fourier Transform) of a sequence, with the only change of replacing ω_n with ω_n^{-1}.

More on DFT

```
1: function FFT*(A)
2:     n ← length[A]
3:     if n = 1 then return A
4:     else
5:         A^[0] ← (A_0, A_2, ...
6:         A^[1] ← (A_1, A_3, ...
7:         y^[0] ← FFT*(A^[0]);
8:         y^[1] ← FFT*(A^[1]);
9:         ω_n ← e^{-i 2π/n};
10:        ω ← 1;
11:        for k = 0 to k = n/2 − 1 do;
12:            y_k ← y_k^[0] + ω · y_k^[1];
13:            y_{n/2+k} ← y_k^[0] − ω · y_k^[1];
14:            ω ← ω · ω_n;
15:        end for
16:        return y;
17:    end if
18: end function
```

```
1: function IFFT*(A)
2:     n ← length[A]
3:     if n = 1 then return A
4:     else
5:         A^[0] ← (A_0, A_2, ... A_{n−2});
6:         A^[1] ← (A_1, A_3, ... A_{n−1});
7:         y^[0] ← IFFT*(A^[0]);
8:         y^[1] ← IFFT*(A^[1]);
9:         ω_n ← e^{i 2π/n};
10:        ω ← 1;
11:        for k = 0 to k = n/2 − 1 do;
12:            y_k ← y_k^[0] + ω · y_k^[1];
13:            y_{n/2+k} ← y_k^[0] − ω · y_k^[1]
14:            ω ← ω · ω_n;
15:        end for
16:        return y;
17:    end if
18: end function
```

```
1: function FFT(A)
2:     return FFT*(A)/√length(A);
3: end function
```

```
1: function IFFT(A)
2:     return IFFT*(A)/√length(A);
3: end function
```

More common form of FFT/IFFT

```
1: function FFT(A)
2:     n ← length[A]
3:     if n = 1 then return A
4:     else
5:         A^[0] ← (A_0, A_2, ...
6:         A^[1] ← (A_1, A_3, ...
7:         y^[0] ← FFT(A^[0]);
8:         y^[1] ← FFT(A^[1]);
9:         ω_n ← e^{-i 2π/n};
10:        ω ← 1;
11:        for k = 0 to k = n/2 − 1 do;
12:            y_k ← y_k^[0] + ω · y_k^[1];
13:            y_{n/2+k} ← y_k^[0] − ω · y_k^[1]
14:            ω ← ω · ω_n;
15:        end for
16:        return y;
17:    end if
18: end function
```

```
1: function IFFT*(A)
2:     n ← length[A]
3:     if n = 1 then return A
4:     else
5:         A^[0] ← (A_0, A_2, ... A_{n−2});
6:         A^[1] ← (A_1, A_3, ... A_{n−1});
7:         y^[0] ← IFFT*(A^[0]);
8:         y^[1] ← IFFT*(A^[1]);
9:         ω_n ← e^{i 2π/n};
10:        ω ← 1;
11:        for k = 0 to k = n/2 − 1 do;
12:            y_k ← y_k^[0] + ω · y_k^[1];
13:            y_{n/2+k} ← y_k^[0] − ω · y_k^[1]
14:            ω ← ω · ω_n;
15:        end for
16:        return y;
17:    end if
18: end function
```

```
1: function IFFT(A)
2:     return IFFT*(A)/length(A);
3: end function
```

Linear Convolution

- Let $A = \langle A_0, A_1, \ldots, A_{n-1} \rangle$ and $B = \langle B_0, B_1, \ldots, B_{m-1} \rangle$ be two sequences of real or complex numbers.
- So A is of length n and B is of length m.
- We can now form two polynomials $P_A(x)$ and $P_B(x)$ with coefficients given by sequences A and B:

$$P_A(x) = \sum_{i=0}^{n-1} A_i \, x^i$$

$$P_B(x) = \sum_{j=0}^{m-1} B_j \, x^j$$

- The product of these two polynomials is a polynomial $P_C(x)$ given by

$$P_C(x) = \sum_{k=0}^{m+n-2} \underbrace{\left(\sum_{i+j=k} A_i B_j \right)}_{C_k} x^k = \sum_{k=0}^{m+n-2} C_k x^k = P_A(x) \cdot P_B(x)$$

Linear Convolution

- Let $A = \langle A_0, A_1, \ldots, A_{n-1} \rangle$ and $B = \langle B_0, B_1, \ldots, B_{m-1} \rangle$ be two sequences of real or complex numbers.
- So A is of length n and B is of length m.
- We can now form two polynomials $P_A(x)$ and $P_B(x)$ with coefficients given by sequences A and B:

$$P_A(x) = \sum_{i=0}^{n-1} A_i \, x^i$$

$$P_B(x) = \sum_{j=0}^{m-1} B_j \, x^j$$

- The product of these two polynomial $P_C(x)$ given by

$$P_C(x) = \sum_{k=0}^{m+n-2} \underbrace{\left(\sum_{i+j=k} A_i B_j \right)}_{C_k} x^k = \sum_{k=0}^{m+n-2} C_k x^k = P_A(x) \cdot P_B(x)$$

Linear Convolution

- Let $A = \langle A_0, A_1, \ldots, A_{n-1} \rangle$ and $B = \langle B_0, B_1, \ldots, B_{m-1} \rangle$ be two sequences of real or complex numbers.
- So A is of length n and B is of length m.
- We can now form two polynomials $P_A(x)$ and $P_B(x)$ with coefficients given by sequences A and B:

$$P_A(x) = \sum_{i=0}^{n-1} A_i \, x^i$$

$$P_B(x) = \sum_{j=0}^{m-1} B_j \, x^j$$

- The product of these two polynomials is a polynomial $P_C(x)$ given by

$$P_C(x) = \sum_{k=0}^{m+n-2} \underbrace{\left(\sum_{i+j=k} A_i B_j \right)}_{C_k} x^k = \sum_{k=0}^{m+n-2} C_k x^k = P_A(x) \cdot P_B(x)$$

Linear Convolution

- Let $A = \langle A_0, A_1, \ldots, A_{n-1} \rangle$ and $B = \langle B_0, B_1, \ldots, B_{m-1} \rangle$ be two sequences of real or complex numbers.
- So A is of length n and B is of length m.
- We can now form two polynomials $P_A(x)$ and $P_B(x)$ with coefficients given by sequences A and B:

$$P_A(x) = \sum_{i=0}^{n-1} A_i \, x^i$$

$$P_B(x) = \sum_{j=0}^{m-1} B_j \, x^j$$

- The product of these two polynomials is a polynomial $P_C(x)$ given by

$$P_C(x) = \sum_{k=0}^{m+n-2} \underbrace{\left(\sum_{i+j=k} A_i B_j \right)}_{C_k} x^k = \sum_{k=0}^{m+n-2} C_k x^k = P_A(x) \cdot P_B(x)$$

Linear Convolution

- The sequence of the coefficients $C = \langle C_0, C_1, \ldots, C_{m+n-2} \rangle$ of the product polynomial $P_C(x)$,

$$P_A(x) \cdot P_B(x) = \sum_{k=0}^{m+n-2} C_k x^k = \sum_{k=0}^{m+n-2} \Bigg(\underbrace{\sum_{i+j=k} A_i B_j}_{C_k} \Bigg) x^k$$

i.e., the sequence

$$\left\langle \sum_{i+j=k} A_i B_j \right\rangle_{k=0}^{m+n-2}$$

of length $m+n-1$ is called *the linear convolution* of sequences A and B.

- If we multiply polynomials $P_A(x)$ and $P_B(x)$ in the usual way we would need to compute $m \times n$ products of the form $A_i B_j$ for all $0 \leq i \leq n-1$ and all $0 \leq j \leq m-1$.
- However, we can use the FFT algorithm to compute the linear convolution of these two sequences in time of only $O\left((m+n)\log_2(m+n)\right)$.

Linear Convolution

- The sequence of the coefficients $C = \langle C_0, C_1, \ldots, C_{m+n-2} \rangle$ of the product polynomial $P_C(x)$,

$$P_A(x) \cdot P_B(x) = \sum_{k=0}^{m+n-2} C_k x^k = \sum_{k=0}^{m+n-2} \Bigg(\underbrace{\sum_{i+j=k} A_i B_j}_{C_k} \Bigg) x^k$$

i.e., the sequence

$$\left\langle \sum_{i+j=k} A_i B_j \right\rangle_{k=0}^{k=m+n-2}$$

of length $m + n - 1$ is called the *linear convolution* of sequences A and B.

- If we multiply polynomials $P_A(x)$ and $P_B(x)$ in the usual way we would need to compute $m \times n$ products of the form $A_i B_j$ for all $0 \leq i \leq n-1$ and all $0 \leq j \leq m-1$.

- However, we can use the FFT algorithm to compute the linear convolution of these two sequences in time of only $O\left((m+n)\log_2(m+n)\right)$.

Linear Convolution

- The sequence of the coefficients $C = \langle C_0, C_1, \ldots, C_{m+n-2} \rangle$ of the product polynomial $P_C(x)$,

$$P_A(x) \cdot P_B(x) = \sum_{k=0}^{m+n-2} C_k x^k = \sum_{k=0}^{m+n-2} \Bigg(\underbrace{\sum_{i+j=k} A_i B_j}_{C_k} \Bigg) x^k$$

i.e., the sequence

$$\left\langle \sum_{i+j=k} A_i B_j \right\rangle_{k=0}^{k=m+n-2}$$

of length $m + n - 1$ is called the *linear convolution* of sequences A and B.

- If we multiply polynomials $P_A(x)$ and $P_B(x)$ in the usual way we would need to compute $m \times n$ products of the form $A_i B_j$ for all $0 \leq i \leq n-1$ and all $0 \leq j \leq m-1$.

- However, we can use the FFT algorithm to compute the linear convolution of these two sequences in time of only $O\left((m+n)\log_2(m+n)\right)$.

Linear Convolution

- The sequence of the coefficients $C = \langle C_0, C_1, \ldots, C_{m+n-2} \rangle$ of the product polynomial $P_C(x)$,

$$P_A(x) \cdot P_B(x) = \sum_{k=0}^{m+n-2} C_k x^k = \sum_{k=0}^{m+n-2} \Bigg(\underbrace{\sum_{i+j=k} A_i B_j}_{C_k} \Bigg) x^k$$

i.e., the sequence

$$\left\langle \sum_{i+j=k} A_i B_j \right\rangle_{k=0}^{k=m+n-2}$$

of length $m + n - 1$ is called the *linear convolution* of sequences A and B.

- If we multiply polynomials $P_A(x)$ and $P_B(x)$ in the usual way we would need to compute $m \times n$ products of the form $A_i B_j$ for all $0 \leq i \leq n-1$ and all $0 \leq j \leq m-1$.

- However, we can use the FFT algorithm to compute the linear convolution of these two sequences in time of only $O\left((m+n)\log_2(m+n)\right)$.

Fast multiplication of polynomials

$P_A(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1}$ $\quad P_B(x) = B_0 + B_1 x + \ldots + B_{n-1} x^{m-1}$

$\Downarrow$ FFT $\quad O((m+n)\log(m+n))$ $\qquad\qquad$ $\Downarrow$ FFT $\quad O((m+n)\log(m+n))$

$\{P_A(1), P_A(\omega_{m+n-1}), P_A(\omega_{m+n-1}^2), \ldots, P_A(\omega_{m+n-1}^{m+n-2})\}$; $\quad \{P_B(1), P_B(\omega_{m+n-1}), \ldots, P_B(\omega_{m+n-1}^{m+n-2})\}$

$\Downarrow$ multiplication $\quad O(m+n)$

$\{P_A(1)P_B(1), \quad P_A(\omega_{m+n-1})P_B(\omega_{m+n-1}), \ldots, P_A(\omega_{m+n-1}^{m+n-2})P_B(\omega_{m+n-1}^{m+n-2})\}$

$\Downarrow$ IFFT $\quad O((m+n)\log(m+n))$

$$P_C(x) = \sum_{k=0}^{m+n-2} \underbrace{\left(\sum_{i+j=k} A_i B_j \right)}_{C_k} x^k = \sum_{k=0}^{m+n-2} C_k x^k = P_A(x) \cdot P_B(x)$$

Thus, the product $P_C(x) = P_A(x)\, P_B(x)$ of two polynomials $P_A(x)$ and $P_B(x)$ of degree n can be computed in time $O(n \log n)$.

Computing the convolution $C = A * B$

$A = \langle A_0, A_1, \ldots, A_{n-1} \rangle$ $\qquad$ $B = \langle B_0, B_1, \ldots, B_{n-1} \rangle$

$\Downarrow \quad O(m+$ $\qquad$ $\Downarrow \quad O(m+n)$

$A^* = \langle A_0, \ldots, A_{n-1}, \underbrace{0, \ldots}_{m+n-1} \rangle$ $\qquad$ $B^* = \langle B_0, \ldots, B_{n-1}, \underbrace{0, \ldots, 0}_{m+n-1} \rangle$

$\Downarrow$ FFT $\quad O((m+n)\log(m+n))$ $\qquad$ $\Downarrow$ FFT

$\langle \widehat{A}_0, \widehat{A}_1, \ldots, \widehat{A}_{m+n-2} \rangle$ $\qquad$ $\langle \widehat{B}_0, \widehat{B}_1, \ldots, \widehat{B}_{m+n-2} \rangle$

$\Downarrow$ multiplication $\quad O(m+n)$

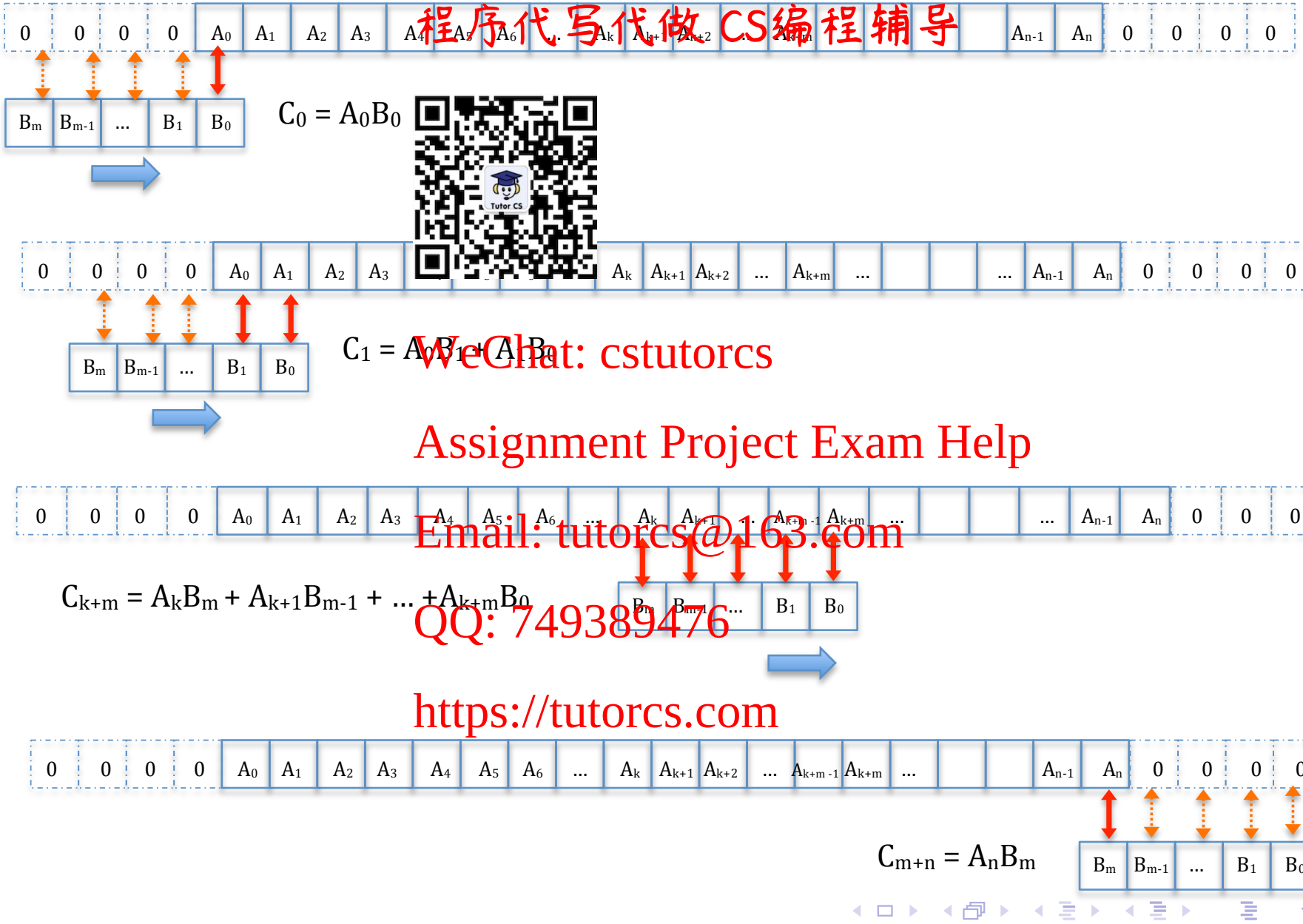
$\langle \widehat{A}_0 \widehat{B}_0, \widehat{A}_1 \widehat{B}_1, \ldots, \widehat{A}_{m+n-2} \widehat{B}_{m+n-2} \rangle$

$\Downarrow$ IFFT $\quad O((m+n)\log(m+n))$

$$C = \left\langle \sum_{i+j=k} A_i B_j \right\rangle_{k=0}^{k=m+n-2}$$

Convolution $C = A * B$ of sequences A and B is computed in time $O((m+n)\log(m+n))$.

Visualizing Convolution $C = A * B$



Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given binary strings A and B; A has n^2 bits, the other, shorter string has n bits.
- Your task is to find all strings of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ everywhere in both A and B with -1, to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \le i \le n-1$).
- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \le m \le 2n-2\}$.

Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given binary strings A and B; A has n^2 bits, the other, shorter string has n bits.
- Your task is to find all strings of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ everywhere in both A and B with -1, to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \le i \le n-1$).
- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \le m \le 2n-2\}$.

Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given binary strings A and B; A has n^2 bits, the other, shorter string B has n bits.
- Your task is to find a substrings of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ everywhere in both A and B with -1, to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \le i \le n-1$).
- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \le m \le 2n-2\}$.

Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given binary strings A and B; A has n^2 bits, the other, shorter string n bits.
- Your task is to find a [illegible]gs of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ [illegible] to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \le i \le n-1$).
- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \le m \le 2n-2\}$.

Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given binary strings A and B; A has n^2 bits, the other, shorter string B has n bits.
- Your task is to find a substrings of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ everywhere in both A and B with -1, to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \leq i \leq n-1$).

- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \leq m \leq 2n-2\}$.

Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given two binary strings A and B; A has n^2 bits, the other, shorter string B has n bits.
- Your task is to find a substrings of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ everywhere in both A and B with -1, to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \leq i \leq n-1$).

- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \leq m \leq 2n-2\}$.

Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given two binary strings A and B; A has n^2 bits, the other, shorter string B has n bits.
- Your task is to find a substrings of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ everywhere in both A and B with -1, to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \leq i \leq n-1$).

- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \leq m \leq 2n-2\}$.

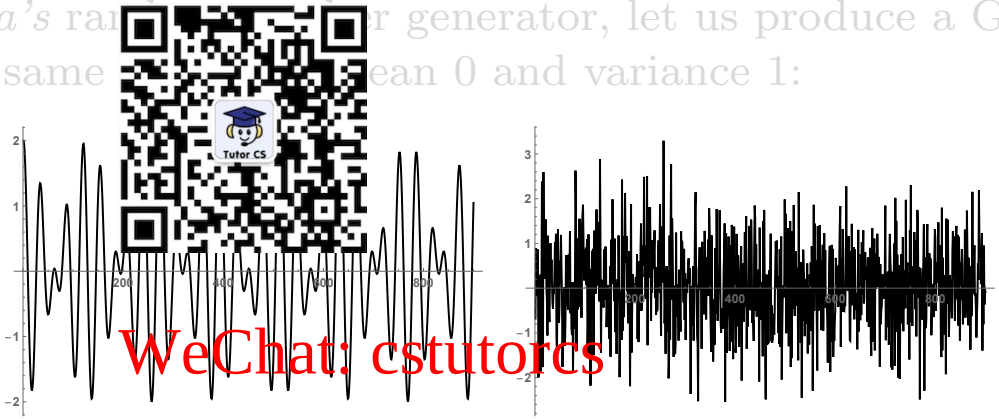
Applications of Convolution

- The degree of similarity of two strings A and B of the same length n can be taken to be the number of places i where $A[i] = B[i]$.
- Assume you are given two binary strings A and B; A has n^2 bits, the other, shorter string has n bits.
- Your task is to find a substrings of consecutive bits in A which are of length n that are the most similar to string B.
- Your algorithm should run in time $O(n^2 \log n)$. Note that the brute force algorithm runs in time $O(n^3)$.
- *Hint:* Replace $0's$ everywhere in both A and B with -1, to obtain sequences A' and B'. Look at the convolution $C = A' * \widetilde{B}'$ where $\widetilde{B}'$ is the mirror image of B' (i.e., $\widetilde{B}'(i) = B'(n-1-i)$ for all $0 \leq i \leq n-1$).

- In this way we are computing the scalar product of B with all the contiguous substrings of A of the same length as B.
- Such scalar product adds one for each pair of matching bits and subtracts one for each pair which is a mismatch.
- Thus, we have to find indices j such that $C(j) = \max\{C(m) : 0 \leq m \leq 2n-2\}$.

Applications of Convolution – Moving Average Smoothing

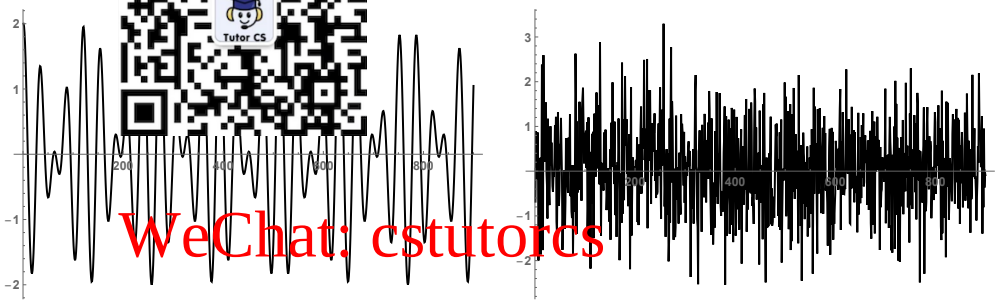
- Let us define a simple signal by $s(t) = \cos\left(\frac{2\pi \cdot 26}{1024} t\right) + \cos\left(\frac{2\pi \cdot 34}{1024} t\right)$, and compute its values for $t = 1 \ldots 900$, thus obtaining sequence $\langle s(i) \ : \ 1 \leq i \leq 900 \rangle$.
- Using *Mathematica's* random number generator, let us produce a Gaussian white noise signal of the same length, with mean 0 and variance 1:



- We add noise to the signal to obtain a noisy signal $\langle n(i) \ : \ 1 \leq i \leq 900 \rangle$; notice that it is now impossible to tell that the underlying signal was a sum of two sine waves:

Applications of Convolution – Moving Average Smoothing

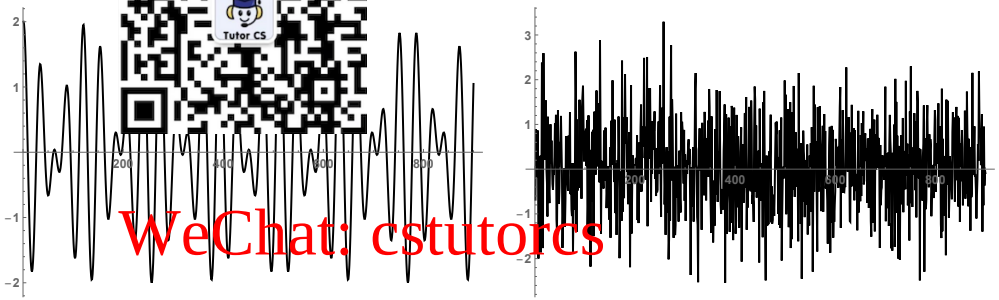
- Let us define a simple signal by $s(t) = \cos\left(\frac{2\pi \cdot 26}{1024} t\right) + \cos\left(\frac{2\pi \cdot 34}{1024} t\right)$, and compute its values for $t = 1 \ldots 900$, thus obtaining sequence $\langle s(i) \ : \ 1 \leq i \leq 900 \rangle$.
- Using *Mathematica's* random number generator, let us produce a Gaussian white noise signal of the same length, mean 0 and variance 1:



- We add noise to the signal to obtain a noisy signal $\langle n(i) \ : \ 1 \leq i \leq 900 \rangle$; notice that it is now impossible to tell that the underlying signal was a sum of two sine waves:

Applications of Convolution – Moving Average Smoothing

- Let us define a simple signal by $s(t) = \cos\left(\frac{2\pi \cdot 26}{1024} t\right) + \cos\left(\frac{2\pi \cdot 34}{1024} t\right)$, and compute its values for $t = 1 \ldots 900$, thus obtaining sequence $\langle s(i) \ : \ 1 \leq i \leq 900 \rangle$.
- Using *Mathematica's* random number generator, let us produce a Gaussian white noise signal of the same length, mean 0 and variance 1:



- We add noise to the signal to obtain a very noisy signal $\langle ns(i) \ : \ 1 \leq i \leq 900 \rangle$; notice that it is now impossible to tell that the underlying signal was a sum of two sine waves:

Applications of Convolution – Moving Average Smoothing

- However, if we plot the absolute value of the DFT of such a noisy signal it is pretty clear that it has tw

- Note that for two sine waves we get 4 peaks. This is because we need two complex exponentials per each cosine wave in order to cancel out the imaginary parts:

$$\cos\left(\frac{2\pi \cdot 26}{1024}\, t\right) = \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\, t} + e^{-i\,\frac{2\pi\cdot 26}{1024}\, t}\right) = \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\, t} + e^{i\,\frac{2\pi\cdot (1024-26)}{1024}\, t}\right)$$
$$= \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\, t} + e^{i\,\frac{2\pi\cdot 998}{1024}\, t}\right)$$

- So $\cos\left(2\pi \cdot 26/1024\, t\right)$ has two peaks: $\widehat{A}_{26}$ and $\widehat{A}_{998}$.

Applications of Convolution – Moving Average Smoothing

- However, if we plot the absolute value of the DFT of such a noisy signal it is pretty clear that it has tw

- Note that for two sine waves we get 4 peaks. This is because we need two complex exponentials per each cosine wave, in order to cancel out the imaginary parts:

$$\cos\left(\frac{2\pi \cdot 26}{1024}\, t\right) = \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\,t} + e^{-i\,\frac{2\pi\cdot 26}{1024}\,t}\right) = \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\,t} + e^{i\,\frac{2\pi\cdot(1024-26)}{1024}\,t}\right)$$
$$= \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\,t} + e^{i\,\frac{2\pi\cdot 998}{1024}\,t}\right)$$

- So $\cos\left(2\pi \cdot 26/1024\, t\right)$ has two peaks: $\widehat{A}_{26}$ and $\widehat{A}_{998}$.

Applications of Convolution – Moving Average Smoothing

- However, if we plot the absolute value of the DFT of such a noisy signal it is pretty clear that it has tw

- Note that for two sine waves we get 4 peaks. This is because we need two complex exponentials per each cosine wave, in order to cancel out the imaginary parts:

$$\cos\left(\frac{2\pi \cdot 26}{1024}\, t\right) = \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\,t} + e^{-i\,\frac{2\pi\cdot 26}{1024}\,t}\right) = \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\,t} + e^{i\,\frac{2\pi\cdot(1024-26)}{1024}\,t}\right)$$
$$= \frac{1}{2}\left(e^{i\,\frac{2\pi\cdot 26}{1024}\,t} + e^{i\,\frac{2\pi\cdot 998}{1024}\,t}\right)$$

- So $\cos\left(2\pi \cdot 26/1024\, t\right)$ has two peaks: $\widehat{A}_{26}$ and $\widehat{A}_{998}$.

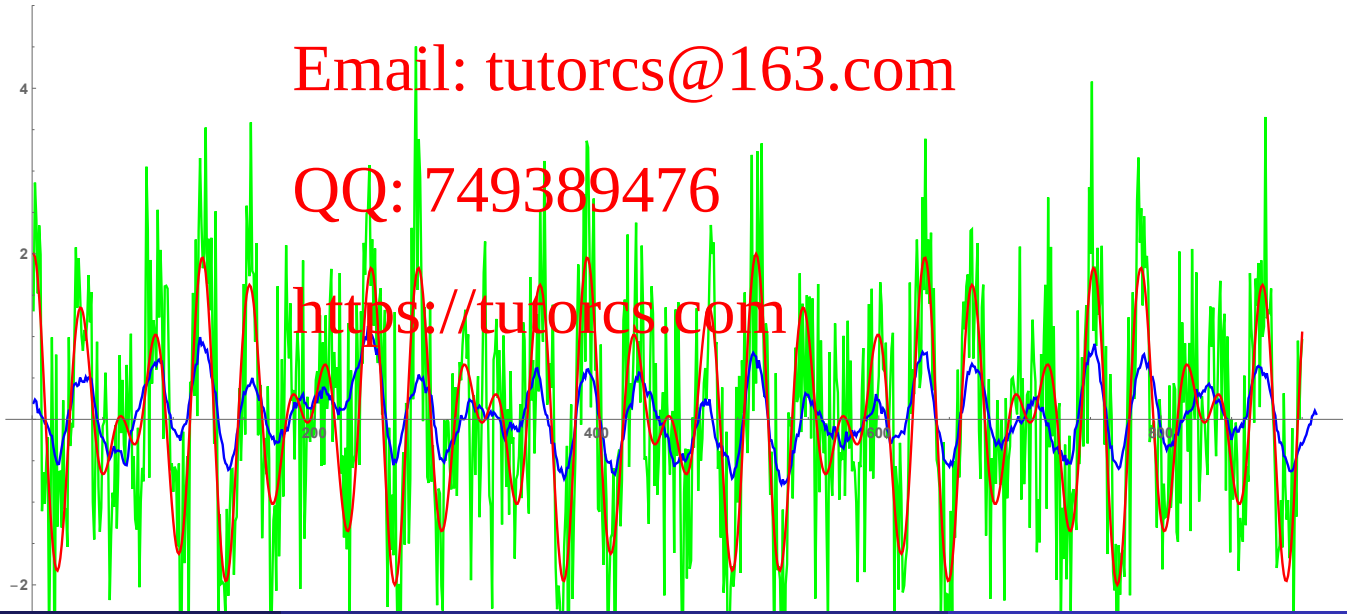
Applications of Convolution – Moving Average Smoothing

- We can smooth such a noisy signal using a *Moving Average*, by replacing every noisy value $ns(i)$ with the mean value of $2M+1$ values of the noisy signal,

$$\frac{1}{2M+1}\sum_{k=-M}^{M} ns(i+k)$$

(setting $ns(i) = 0$ for $i <$ [illegible] > 900).

- The degree of smoothing [illegible] M; larger values of M produce more smoothing but also distort more the underlying "clean signal" $s(i)$.
- Here we choose $M = 10$. The clean signal is in red, the noisy signal in green and the signal smoothed via the Moving Average is in blue; plots are produced using the *Mathematica* software, free for students of UNSW (you can download it from the UNSW IT site).



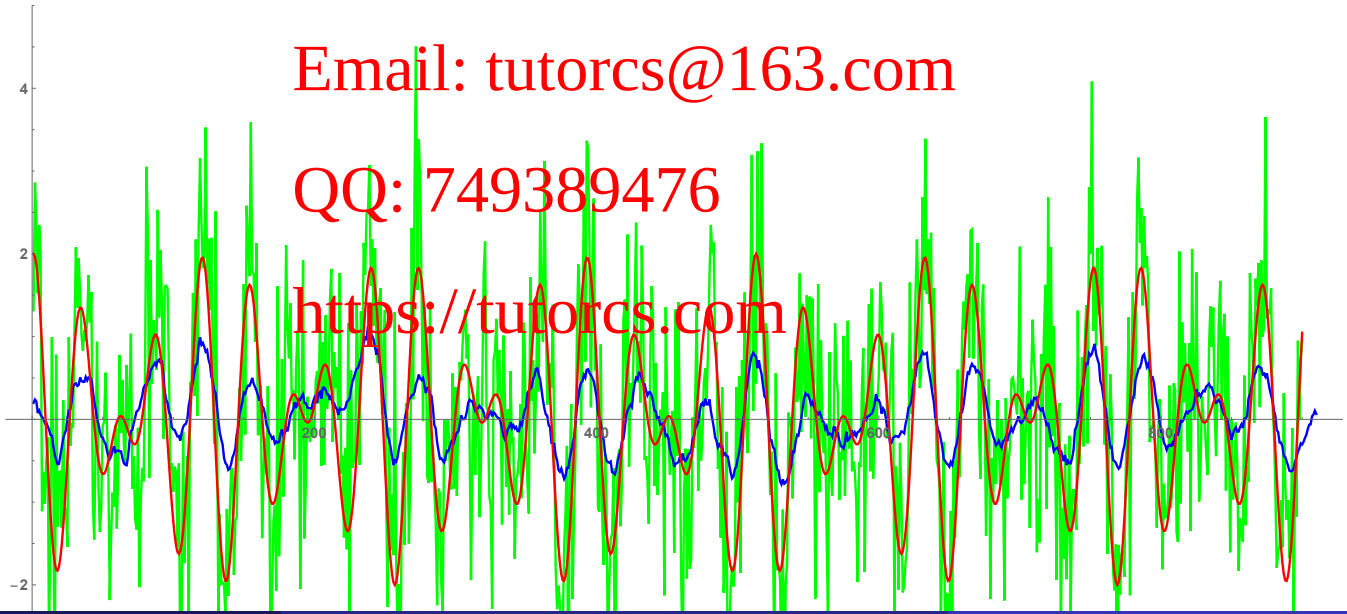
Applications of Convolution – Moving Average Smoothing

- We can smooth such a noisy signal using a *Moving Average*, by replacing every noisy value $ns(i)$ with the mean value of $2M+1$ values of the noisy signal,

$$\frac{1}{2M+1}\sum_{k=-M}^{M} ns(i+k)$$

(setting $ns(i) = 0$ for $i <$ [illegible] > 900).

- The degree of smoothing [illegible] M; larger values of M produce more smoothing but also distort more the underlying "clean signal" $s(i)$.
- Here we choose $M = 10$. The clean signal is in red, the noisy signal in green and the signal smoothed via the Moving Average is in blue; plots are produced using the *Mathematica* software, free for students of UNSW (you can download it from the UNSW IT site).



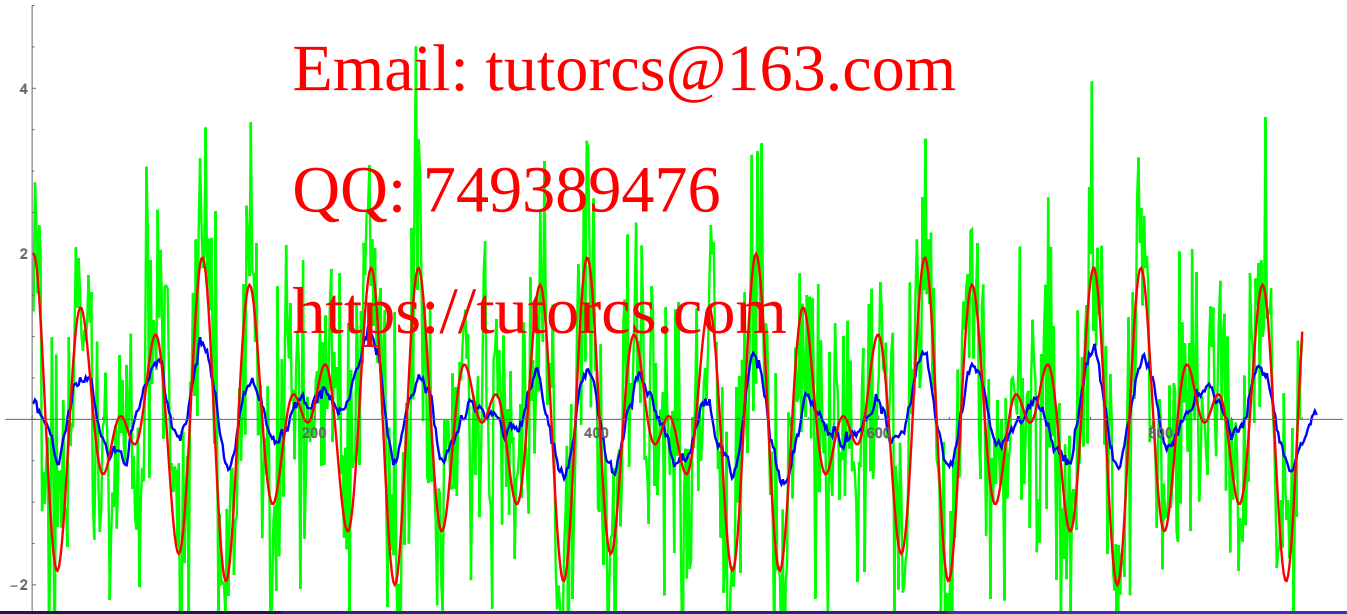
Applications of Convolution – Moving Average Smoothing

- We can smooth such a noisy signal using a *Moving Average*, by replacing every noisy value $ns(i)$ with the mean value of $2M+1$ values of the noisy signal,

$$\frac{1}{2M+1}\sum_{k=-M}^{M} ns(i+k)$$

(setting $ns(i) = 0$ for $i <$ [illegible] > 900).

- The degree of smoothing [illegible] M; larger values of M produce more smoothing but also distort more the underlying "clean signal" $s(i)$.
- Here we choose $M = 10$. The clean signal is in red, the noisy signal in green and the signal smoothed via the Moving Average is in blue; plots are produced using the *Mathematica* software, free for students of UNSW (you can download it from the UNSW IT site).



Applications of Convolution – Moving Average Smoothing

- How do we compute the Moving Average of a signal?

- Applying $ms(i) = \frac{1}{2M+1} \sum_{k=-M}^{M} (i+k)$ directly to a signal with N many data points would involve $(2M+1)N$ additions and N divisions.
- Much faster way, involving only $2M+1+2N$ additions and N divisions computes only the first value $ms(1)$ directly using $2M+1$ additions and one division and then proceeds by recursion by letting $ms(i+1) = ms(i) + (ns(i+1+M) - ns(i-M))/(2M+1)$.
- This is correct because

$$ms(i+1) = \frac{ns(i+1-M) + \ldots + ns(i+1) + \ldots + ns(i+1+M)}{2M+1}$$

$$= \frac{(ns(i-M) + ns(i+1-M) + \ldots + ns(i+1) + \ldots + ns(i+M)) + ns(i+1+M) - ns(i-M)}{2M+1}$$

$$= ms(i) + \frac{ns(i+1+M) - ns(i-M)}{2M+1}$$

Applications of Convolution – Moving Average Smoothing

- How do we compute the Moving Average of a signal?
- Applying $ms(i) = \frac{1}{2M+1}\sum_{k=-M}^{M}(i+k)$ directly to a signal with N many data points would involve $(2M$ additions and N divisions.
- Much faster way, involving only $2M+1+2N$ additions and N divisions computes only the first value $ms(1)$ directly using $2M+1$ additions and one division and then proceeds by recursion by letting
$ms(i+1) = ms(i) + (ns(i+1+M) - ns(i-M))/(2M+1)$
- This is correct because

$$ms(i+1) = \frac{ns(i+1-M) + \ldots + ns(i+1) + \ldots + ns(i+1+M)}{2M+1}$$

$$= \frac{(ns(i-M) + ns(i+1-M) + \ldots + ns(i+1) + \ldots + ns(i+M)) + ns(i+1+M) - ns(i-M)}{2M+1}$$

$$= ms(i) + \frac{ns(i+1+M) - ns(i-M)}{2M+1}$$

Applications of Convolution – Moving Average Smoothing

- How do we compute the Moving Average of a signal?
- Applying $ms(i) = \frac{1}{2M+1}\sum_{k=-M}^{M}(i+k)$ directly to a signal with N many data points would involve $(2M$ additions and N divisions.
- Much faster way, involving only $2M+1+2N$ additions and N divisions computes only the first value $ms(1)$ directly using $2M+1$ additions and one division and then proceeds by recursion by letting
$ms(i+1) = ms(i) + (ns(i+1+M) - ns(i-M))/(2M+1)$.
- This is correct because

$$ms(i+1) = \frac{ns(i+1-M)+\ldots+ns(i+1)+\ldots+ns(i+1+M)}{2M+1}$$

$$= \frac{(ns(i-M)+ns(i+1-M)+\ldots+ns(i+1)+\ldots+ns(i+M))+ns(i+1+M)-ns(i-M)}{2M+1}$$

$$= ms(i) + \frac{ns(i+1+M)-ns(i-M)}{2M+1}$$

Applications of Convolution – Moving Average Smoothing

- How do we compute the Moving Average of a signal?
- Applying $ms(i) = \frac{1}{2M+1}\sum_{k=-M}^{M}(i+k)$ directly to a signal with N many data points would involve $(2M$ additions and N divisions.
- Much faster way, involving only $2M+1+2N$ additions and N divisions computes only the first value $ms(1)$ directly using $2M+1$ additions and one division and then proceeds by recursion by letting
  $ms(i+1) = ms(i) + (ns(i+1+M) - ns(i-M))/(2M+1)$.
- This is correct because

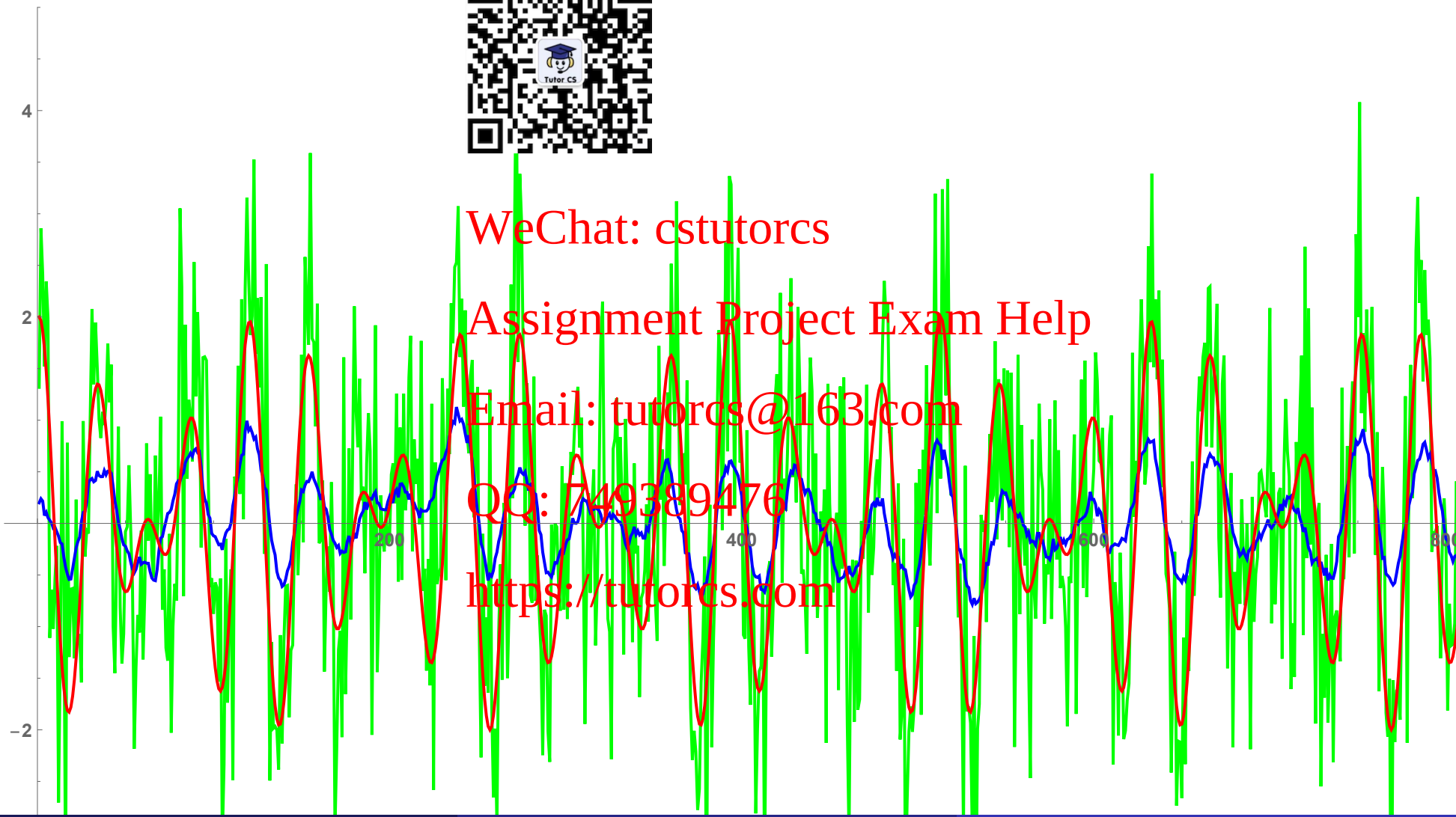
$$ms(i+1) = \frac{ns(i+1-M) + \ldots + ns(i+1) + \ldots + ns(i+1+M)}{2M+1}$$

$$= \frac{(ns(i-M) + ns(i+1-M) + \ldots + ns(i+1) + \ldots + ns(i+M)) + ns(i+1+M) - ns(i-M)}{2M+1}$$

$$= ms(i) + \frac{ns(i+1+M) - ns(i-M)}{2M+1}$$

Applications of Convolution – Moving Average Smoothing

However, the signal obtained by smoothing via a Moving Average does not look like a particularly good replica of the original, uncorrupted signal:



Applications of Convolution – Moving Average Smoothing

- Can we do better?
- Note that the Moving Average Smoothing assigns equal weight of $\frac{1}{2M+1}$ to all of $2M+1$ many samples which are averaged:

$$ms(i) = \frac{1}{2M+1} \sum_{k=-M}^{M} ns(i+k) = \sum_{k=-M}^{M} \frac{1}{2M+1} \cdot ns(i+k)$$

- Maybe there are better weights $w_{-M}, w_{-M+1}, \ldots, 0, \ldots, w_{M-1}, w_M$, which get smaller as you get away of the central point 0, $w_{-M} < w_{-M+1} < \ldots < w_{-1} < w_0 > w_1 > \ldots > w_{M-1} > w_M$?

Applications of Convolution – Moving Average Smoothing

- Can we do better?
- Note that the Moving A  equal weight of $\frac{1}{2M+1}$ to all of $2M+1$ many samples which are averaged:

$$ms(i) = \frac{1}{2M+1} \sum_{k=-M}^{M} ns(i+k) = \sum_{k=-M}^{M} \frac{1}{2M+1} \cdot ns(i+k)$$

- Maybe there are better weights $w_{-M}, w_{-M+1}, \ldots, 0, \ldots, w_{M-1}, w_M$, which get smaller as you get away of the central point 0,
  $w_{-M} < w_{-M+1} < \ldots < w_{-1} < w_0 > w_1 > \ldots > w_{M-1} > w_M$?

Applications of Convolution – Moving Average Smoothing

- Can we do better?
- Note that the Moving A  equal weight of $\frac{1}{2M+1}$ to all of $2M+1$ many samples which are averaged:

$$ms(i) = \frac{1}{2M+1} \sum_{k=-M}^{M} ns(i+k) = \sum_{k=-M}^{M} \frac{1}{2M+1} \cdot ns(i+k)$$

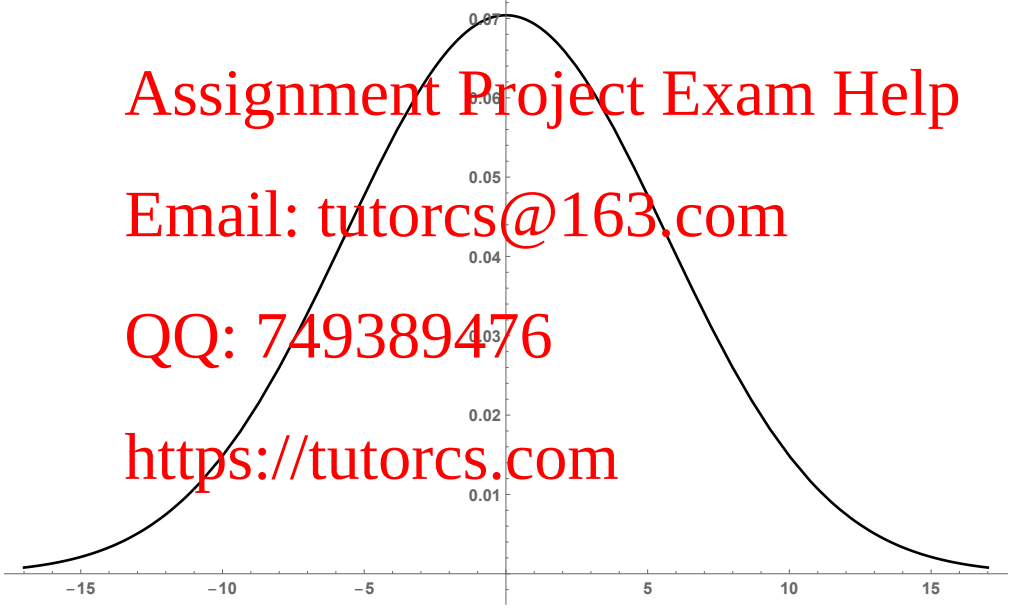
- Maybe there are better weights $w_{-M}, w_{-M+1}, \ldots, 0, \ldots, w_{M-1}, w_M$, which get smaller as you get away of the central point 0, $w_{-M} < w_{-M+1} < \ldots < w_{-1} < w_0 > w_1 > \ldots > w_{M-1} > w_M$?

Applications of Convolution – Gaussian Smoothing

- Gaussian is just the normal distribution with probability density given by

$$) = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{t^2}{2v}}$$

- It "dies off" almost completely whithin the interval $[-3\sqrt{v}, 3\sqrt{v}]$:

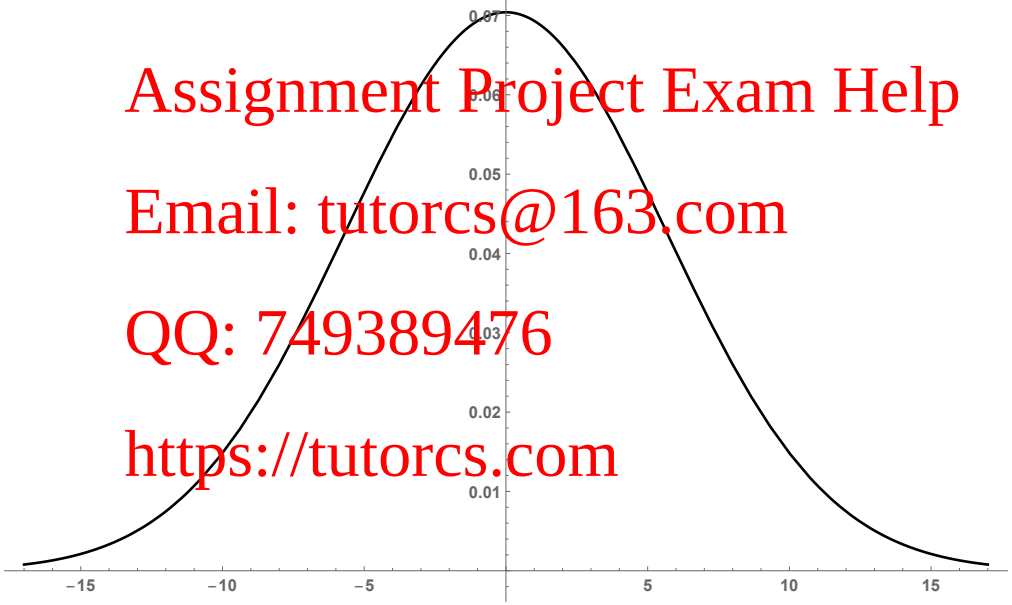


Applications of Convolution – Gaussian Smoothing

- Gaussian is just the normal distribution with probability density given by

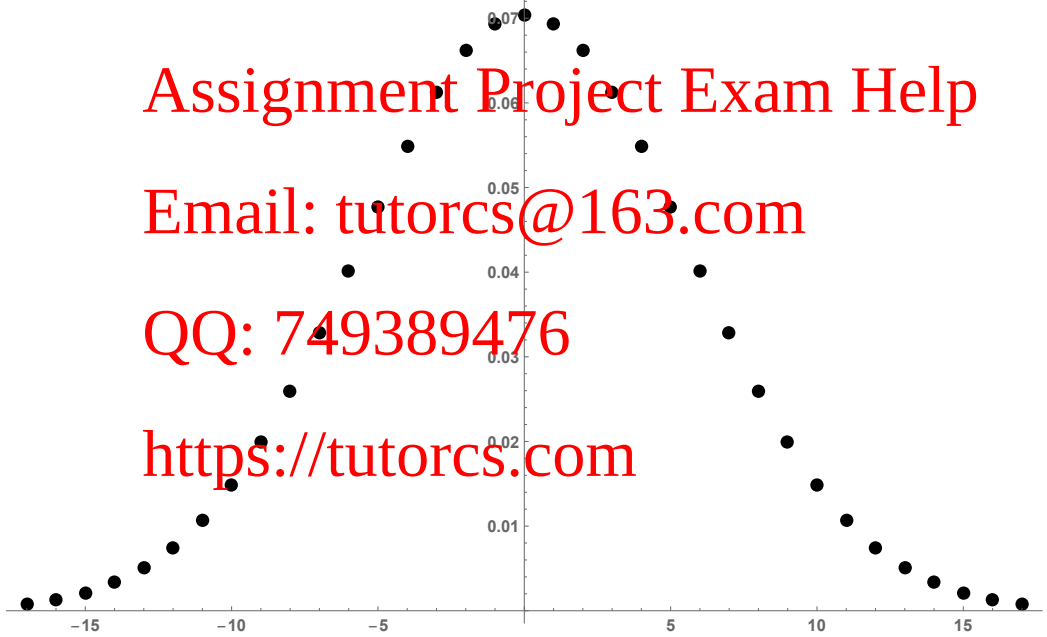
$$) = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{t^2}{2v}}$$

- It "dies off" almost completely whithin the interval $[-3\sqrt{v}, 3\sqrt{v}]$:



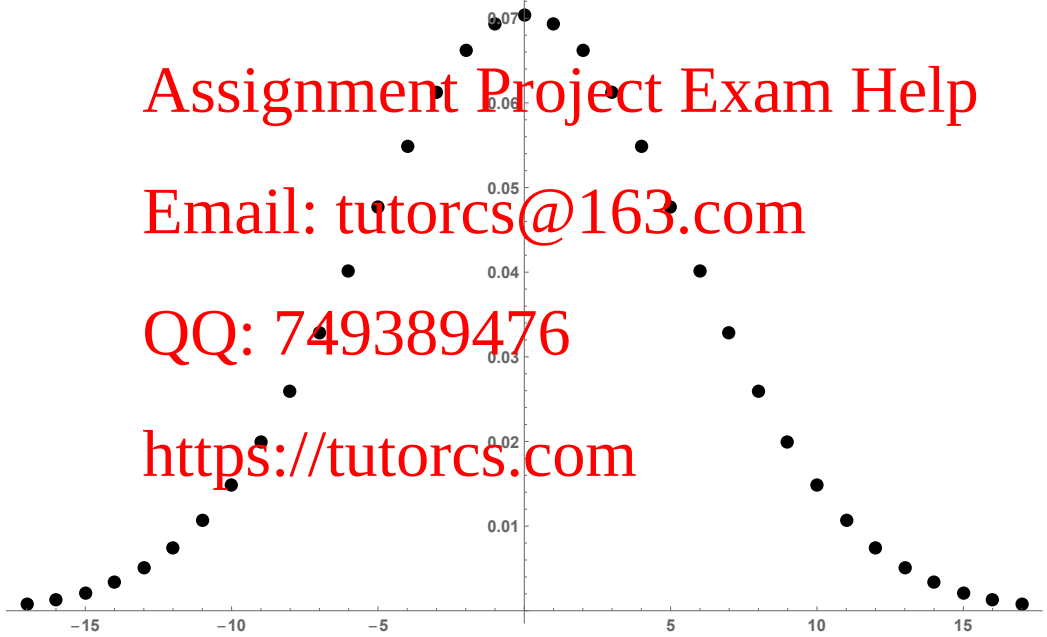
Applications of Convolution – Gaussian Smoothing

- We choose v such that $3\sqrt{v} = 17$ and evaluate such a Gaussian at integers $-17 \ldots 17$ and take these values as weights for our smoothing; thus, $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$
- If k ranges between -17 and $17 = 3\sqrt{v}$, then these weights sum up almost exactly to 1, but you can normalise them to sum up exactly to 1 by replacing each w_k with $w_k / \sum_{m=-17}^{17} w_m$.



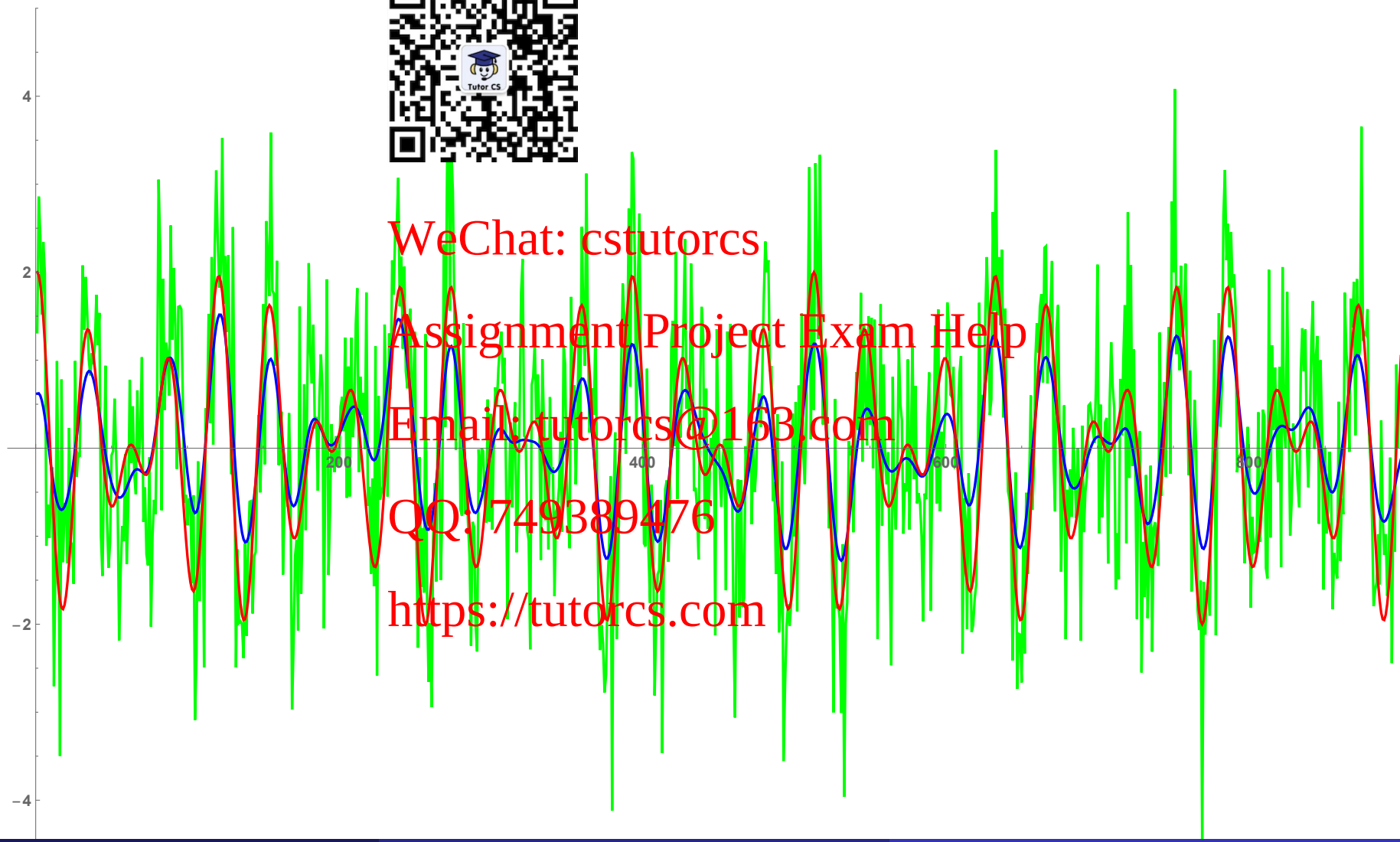
Applications of Convolution – Gaussian Smoothing

- We choose v such that $3\sqrt{v} = 17$ and evaluate such a Gaussian at integers $-17 \ldots 17$ and take these values as weights for our smoothing; thus, $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$
- If k ranges between $-3\sqrt{v}$ and $3\sqrt{v}$, then these weights sum up almost exactly to 1, but you can normalise them to sum up exactly to 1 by replacing each w_k with $w_k / \sum_{m=-17}^{17} w_m$.



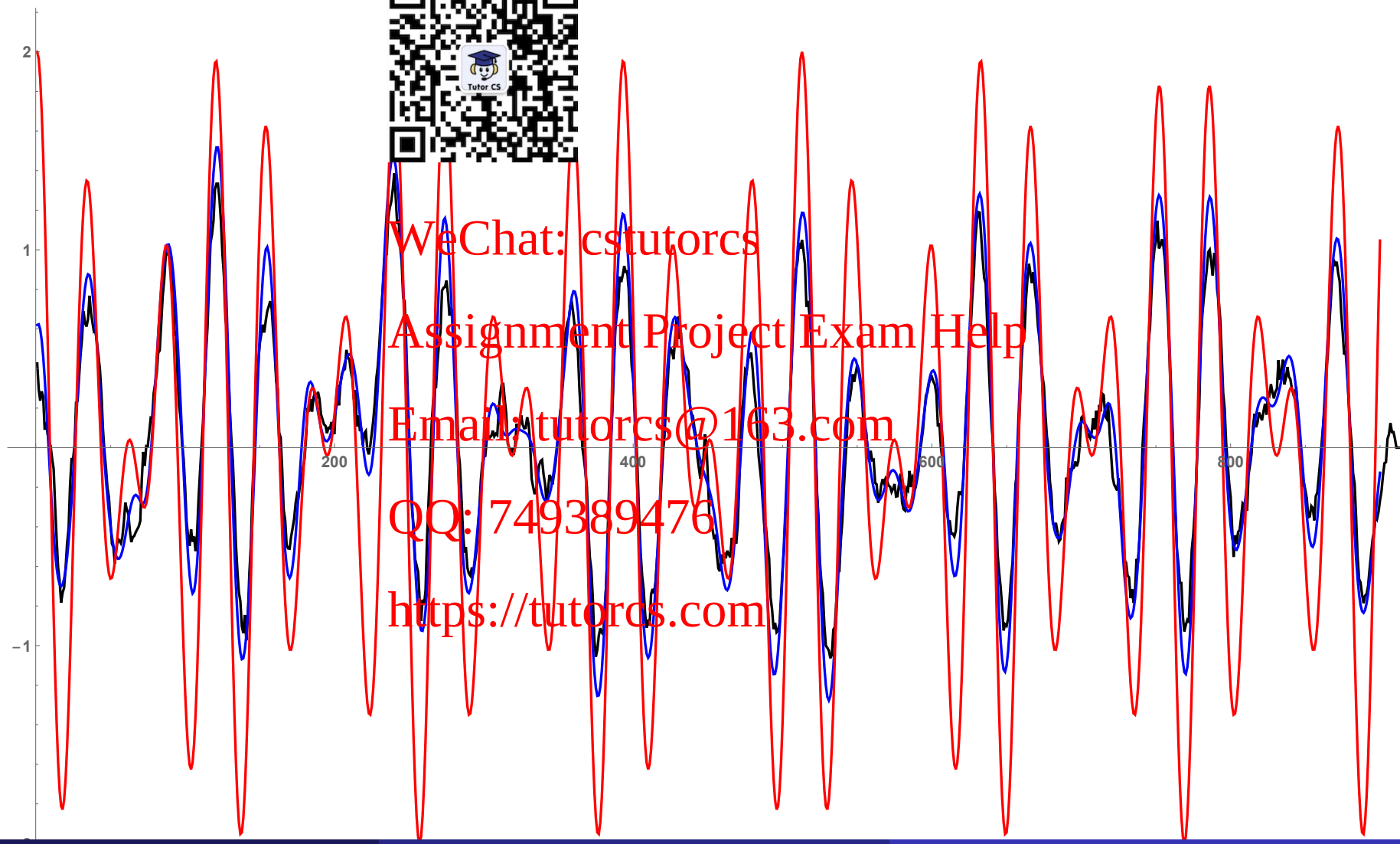
Applications of Convolution – Gaussian Smoothing

- We now produce a smoothing $gs(i)$ of the noisy signal $ns(i)$ by letting $gs(i) = \sum_{k=-17}^{17} w_k \, ns(i-k)$:



Applications of Convolution – Gaussian Smoothing

- We now compare the Moving Average smoothing (black) of our noisy signal with its Gaussian smoothing (blue); the clean signal is shown in red:



Applications of Convolution – Gaussian Smoothing

- We see that the Gaussian smoothing produces better results. But how do we compute it efficiently?

- Since the weights are no[rmalized, we can use the] old trick of computing the smoothing at point $i+$ from the smoothed valu[e at point $i-1$ by] dropping the first term of the sum and adding the new term clearly no[w, and applying the formula] $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i$[to all values of the noisy signal would result in about] $(2M+1)N$ multiplications.
- However, notice that $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i-k)$ is simply a convolution of the noisy signal with the vector of weights! And since the weights are symmetric, we do not even have to flip them!
- Thus, the Gaussian smoothing is simply the convolution $ns * W$ where $W = \langle w_{-M}, w_{-M+1}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{M-1}, w_M \rangle$ with $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$ where the variance v is chosen so that $M = \lceil 3\sqrt{v} \rceil$, and ns is the sequence of N noisy samples of the signal we want to smooth.
- If we evaluate the convolution using the FFT, then we make only $O((M+N)\log_2(M+N))$ multiplications, which is faster than $(2M+1)N$ multiplications of the brute force.
- All of the above calculations can be found in a Mathematica file "A Mathematica implementation of Gaussian smoothing" at the course website.

Applications of Convolution – Gaussian Smoothing

- We see that the Gaussian smoothing produces better results. But how do we compute it efficiently?
- Since the weights are no old trick of computing the smoothing at point $i+$ from the smoothed value by dropping the first term of the sum and adding the new term clearly no s, and applying the formula $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i$ alues of the noisy signal would result in about $(2M+1)N$ multiplications.
- However, notice that $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i+k)$ is simply a convolution of the noisy signal with the vector of weights! And since the weights are symmetric, we do not even have to flip them.
- Thus, the Gaussian smoothing is simply the convolution $ns * W$ where $W = \langle w_{-M}, w_{-M+1}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{M-1}, w_M \rangle$ with $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$ where the variance v is chosen so that $M = \lceil 3\sqrt{v} \rceil$, and ns is the sequence of N noisy samples of the signal we want to smooth.
- If we evaluate the convolution using the FFT, then we make only $O((M+N)\log_2(M+N))$ multiplications, which is faster than $(2M+1)N$ multiplications of the brute force.
- All of the above calculations can be found in a Mathematica file "A Mathematica implementation of Gaussian smoothing" at the course website.

Applications of Convolution – Gaussian Smoothing

- We see that the Gaussian smoothing produces better results. But how do we compute it efficiently?
- Since the weights are no[illegible]old trick of computing the smoothing at point $i+$ from the smoothed value[illegible]by dropping the first term of the sum and adding the new term clearly no[illegible]s, and applying the formula $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i$[illegible]values of the noisy signal would result in about $(2M+1)N$ multiplications.
- However, notice that $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i-k)$ is simply a convolution of the noisy signal with the vector of weights! And since the weights are symmetric, we do not even have to flip them.
- Thus, the Gaussian smoothing is simply the convolution $ns * W$ where $W = \langle w_{-M}, w_{-M+1}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{M-1}, w_M \rangle$ with $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$ where the variance v is chosen so that $M = \lceil 3\sqrt{v} \rceil$, and ns is the sequence of N noisy samples of the signal we want to smooth.
- If we evaluate the convolution using the FFT, then we make only $O((M+N)\log_2(M+N))$ multiplications, which is faster than $(2M+1)N$ multiplications of the brute force.
- All of the above calculations can be found in a Mathematica file "A Mathematica implementation of Gaussian smoothing" at the course website.

Applications of Convolution – Gaussian Smoothing

- We see that the Gaussian smoothing produces better results. But how do we compute it efficiently?
- Since the weights are no [illegible] old trick of computing the smoothing at point $i+$ from the smoothed value [illegible] by dropping the first term of the sum and adding the new term clearly no [illegible]s, and applying the formula $gs(i) = \sum_{k=-M}^{M} w_k \, ns(i$ [illegible] values of the noisy signal would result in about $(2M+1)N$ multiplications.
- However, notice that $gs(i) = \sum_{k=-M}^{M} w_k \, ns(i-k)$ is simply a convolution of the noisy signal with the vector of weights! And since the weights are symmetric, we do not even have to flip them.
- Thus, the Gaussian smoothing is simply the convolution $ns * W$ where $W = \langle w_{-M}, w_{-M+1}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{M-1}, w_M \rangle$ with $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$ where the variance v is chosen so that $M = \lceil 3\sqrt{v} \rceil$, and ns is the sequence of N noisy samples of the signal we want to smooth.
- If we evaluate the convolution using the FFT, then we make only $O((M+N)\log_2(M+N))$ multiplications, which is faster than $(2M+1)N$ multiplications of the brute force.
- All of the above calculations can be found in a Mathematica file "A Mathematica implementation of Gaussian smoothing" at the course website.

Applications of Convolution – Gaussian Smoothing

- We see that the Gaussian smoothing produces better results. But how do we compute it efficiently?
- Since the weights are no old trick of computing the smoothing at point $i+$ from the smoothed value y dropping the first term of the sum and adding the new term clearly no s, and applying the formula $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i$ alues of the noisy signal would result in about $(2M+1)N$ multiplications.
- However, notice that $gs(i) = \sum_{k=-M}^{M} w_k ns(i-k)$ is simply a convolution of the noisy signal with the vector of weights! And since the weights are symmetric, we do not even have to flip them.
- Thus, the Gaussian smoothing is simply the convolution $ns * W$ where $W = \langle w_{-M}, w_{-M+1}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{M-1}, w_M \rangle$ with $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$ where the variance v is chosen so that $M = \lceil 3\sqrt{v} \rceil$, and ns is the sequence of N noisy samples of the signal we want to smooth.
- If we evaluate the convolution using the FFT, then we make only $O((M+N)\log_2(M+N))$ multiplications, which is faster than $(2M+1)N$ multiplications of the brute force.
- All of the above calculations can be found in a Mathematica file "A Mathematica implementation of Gaussian smoothing" at the course website.

Applications of Convolution – Gaussian Smoothing

- We see that the Gaussian smoothing produces better results. But how do we compute it efficiently?
- Since the weights are no old trick of computing the smoothing at point $i+$ from the smoothed value by dropping the first term of the sum and adding the new term clearly no s, and applying the formula $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i$ alues of the noisy signal would result in about $(2M+1)N$ multiplications.
- However, notice that $gs(i) = \sum_{k=-M}^{M} w_k\, ns(i-k)$ is simply a convolution of the noisy signal with the vector of weights! And since the weights are symmetric, we do not even have to flip them.
- Thus, the Gaussian smoothing is simply the convolution $ns * W$ where $W = \langle w_{-M}, w_{-M+1}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{M-1}, w_M \rangle$ with $w_k = \frac{1}{\sqrt{2\pi\, v}} e^{-\frac{k^2}{2v}}$ where the variance v is chosen so that $M = \lceil 3\sqrt{v} \rceil$, and ns is the sequence of N noisy samples of the signal we want to smooth.
- If we evaluate the convolution using the FFT, then we make only $O((M+N)\log_2(M+N))$ multiplications, which is faster than $(2M+1)N$ multiplications of the brute force.
- All of the above calculations can be found in a Mathematica file "A Mathematica implementation of Gaussian smoothing" at the course website.