TUAT Intensive Course
Data Analysis on Graphs and Networks
Day 1: Basics of Data Analysis on Regular Lattices

Naoki Saito

Department of Mathematics
University of California, Davis

As a part of "Green & Clean Food Production Advancement I"
Fuchu Campus, Tokyo University of Agriculture & Technology
August 27, 2014

Outline

1. Motivations
2. Basics and Some History of Fourier Analysis (through the view of 1D Wave Equation)
3. Basics of Data Representation and Compression on Regular Lattice via Linear Algebra

Acknowledgment

- Jeff Irion (UC Davis)
- Risa Naito (TUAT)
- Yuji Nakatsukasa (formerly UC Davis; currently Univ. of Tokyo)
- Kenshi Sakai (TUAT)
- Ernest Woei (formerly UC Davis; currently Flash Foto, Inc.)
- Support from Office of Naval Research grant: ONR N00014-12-1-0177
- Support from National Science Foundation grant: DMS-1418779
- Support for Jeff Irion from National Defense Science and Engineering Graduate Fellowship, 32 CFR 168a via AFOSR FA9550-11-C-0028
- The MacTutor History of Mathematics Archive, Wikipedia, ...

Lecture Outline

Day 1 (13:00-16:15, August 27; Fuchu Campus):

- Motivations; Importance of Data Analysis on Networks and Graphs
- Basics (and Some History) of Fourier Analysis
- Basics of Data Representation and Compression on Regular Lattices via Linear Algebra and Fourier Analysis

Day 2 (13:00-18:00, August 28; Fuchu Campus):

- Basics of Graph Theory, Graph Laplacian Eigenvalues/Eigenvectors
- Graph partitioning
- Multiscale Basis Dictionaries on Graphs and Networks
- Applications (signal denoising, morphological analysis of neuronal dendritic trees, etc.)
- Discussions on potential agricultural applications including "Green and Clean Food Productions" with participants

General Basic References

- R. B. Bapat: *Graphs and Matrices*, Universitext, Springer, 2010.
- F. R. K. Chung: *Spectral Graph Theory*, Amer. Math. Soc., 1997.
- D. Cvetković, P. Rowlinson, & S. Simić: *An Introduction to the Theory of Graph Spectra*, Vol. 75, London Mathematical Society Student Texts, Cambridge Univ. Press, 2010.
- N. Masuda & N. Konno: *Complex Networks*, Kindai-Kagaku-Sha, 2010 (in Japanese).
- D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, & P. Vandergheynst: "The emerging field of signal processing on graphs," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.
- H. Urakawa: *Laplacian & Networks*, Shokabo, 1996 (in Japanese).
- N. Saito's Course on Applied Linear Algebra (http://www.math.ucdavis.edu/~saito/courses/167/lectures.html)
- N. Saito's Course on Harmonic Analysis on Graphs and Networks (http://www.math.ucdavis.edu/~saito/courses/HarmGraph/refs.html)

More specific references are given throughout the lectures.

Outline

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, . . .
  - Data from biological networks
  - . . .
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, ...
  - Data from biological networks
  - ...
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, ...
  - Data from biological networks
  - ...
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, ...
  - Data from biological networks
  - ...
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, ...
  - Data from biological networks
  - ...
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, ...
  - Data from biological networks
  - ...
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, ...
  - Data from biological networks
  - ...
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, . . .
  - Data from biological networks
  - . . .
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs & Networks?

- Fourier analysis/synthesis and wavelet analysis/synthesis have been 'crown jewels' for data sampled on the regular lattices.
- Hence, we need to lift such tools for unorganized and irregularly-sampled datasets including those represented by graphs and networks.
- Moreover, constructing a graph from a usual signal or image and analyzing it can also be very useful! E.g., Nonlocal means image denoising of Buades-Coll-Morel.

Motivations: Why Graphs & Networks?

- Fourier analysis/synthesis and wavelet analysis/synthesis have been ‘crown jewels’ for data sampled on the regular lattices.
- Hence, we need to lift such tools for unorganized and irregularly-sampled datasets including those represented by graphs and networks.
- Moreover, constructing a graph from a usual signal or image and analyzing it can also be very useful! E.g., Nonlocal means image denoising of Buades-Coll-Morel.

Motivations: Why Graphs & Networks?

- Fourier analysis/synthesis and wavelet analysis/synthesis have been 'crown jewels' for data sampled on the regular lattices.
- Hence, we need to lift such tools for unorganized and irregularly-sampled datasets including those represented by graphs and networks.
- Moreover, constructing a graph from a usual signal or image and analyzing it can also be very useful! E.g., Nonlocal means image denoising of Buades-Coll-Morel.

An Example of Sensor Networks

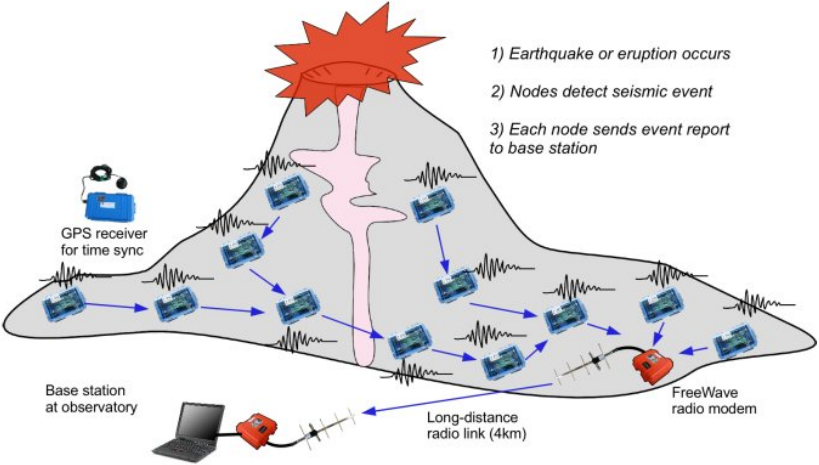


Figure: Volcano monitoring sensor network architecture of Harvard Sensor Networks Lab

An Example of Social Networks

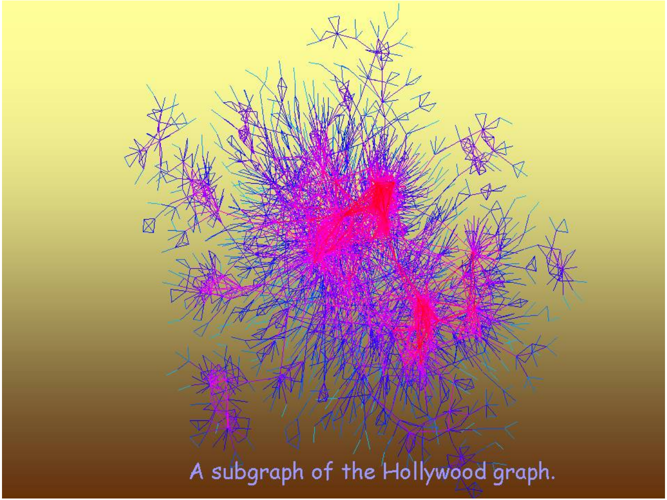


Figure: Through the courtesy of Prof. Fan Chung, UC San Diego

An Example of Biological Networks

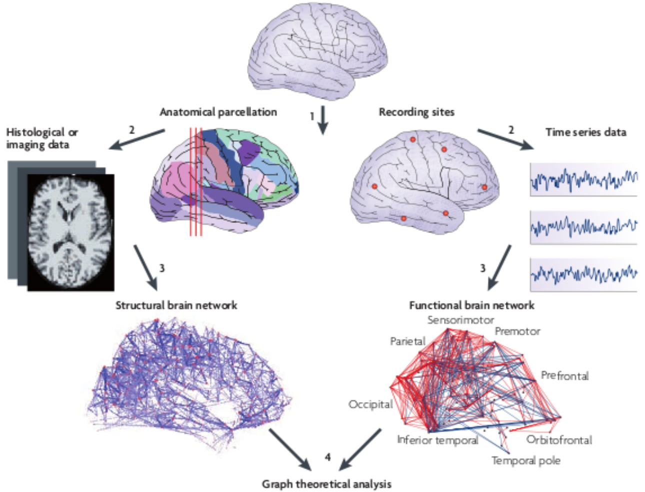
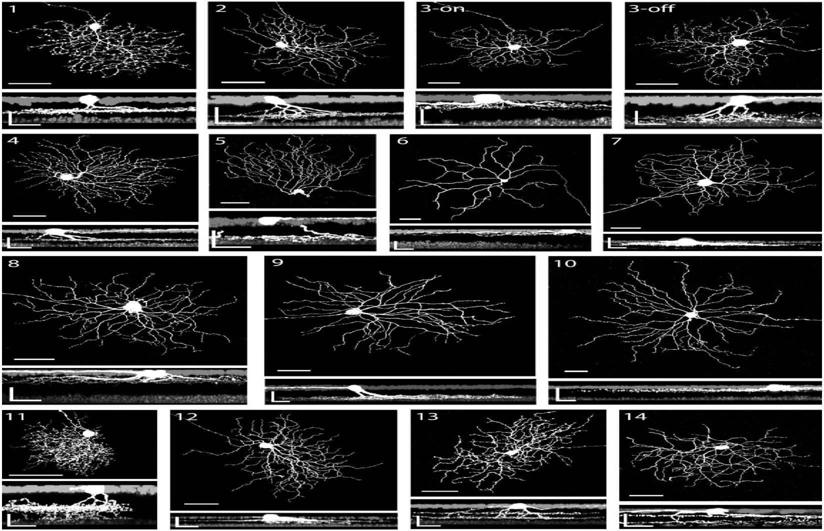
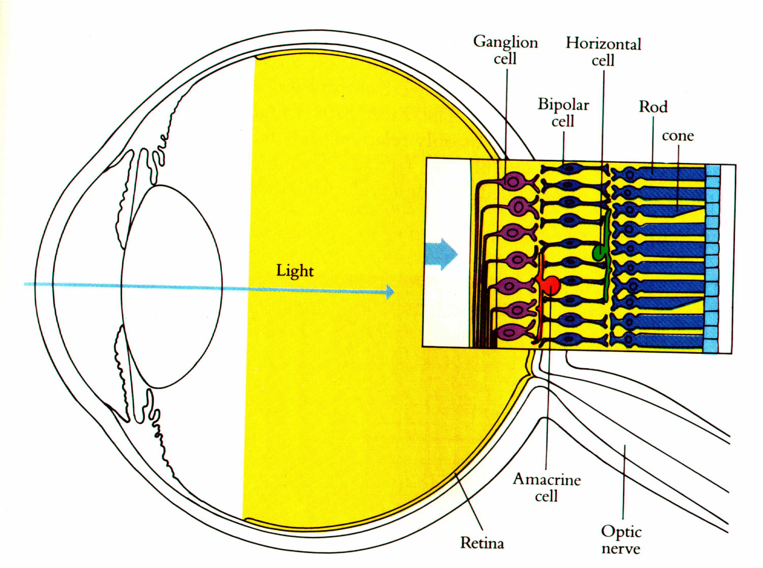


Figure: From E. Bullmore and O. Sporns, *Nature Reviews Neuroscience*, vol. 10, pp.186–198, Mar. 2009.

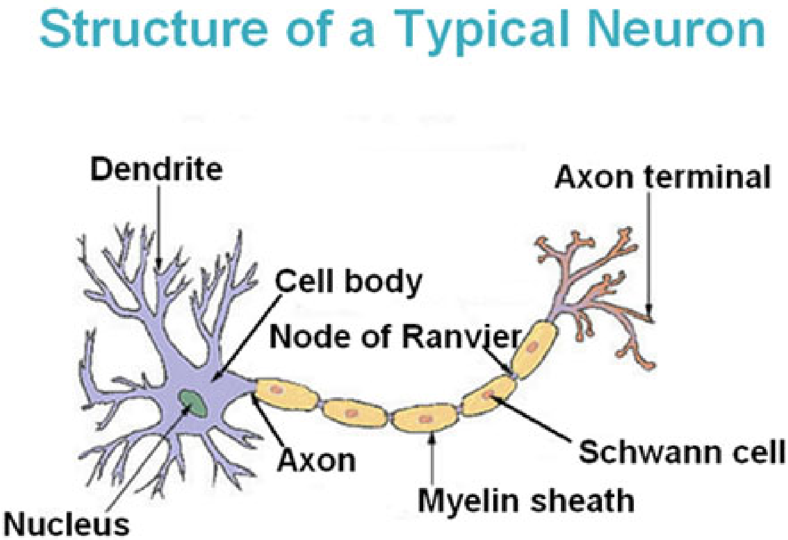
Another Biological Example: Retinal Ganglion Cells



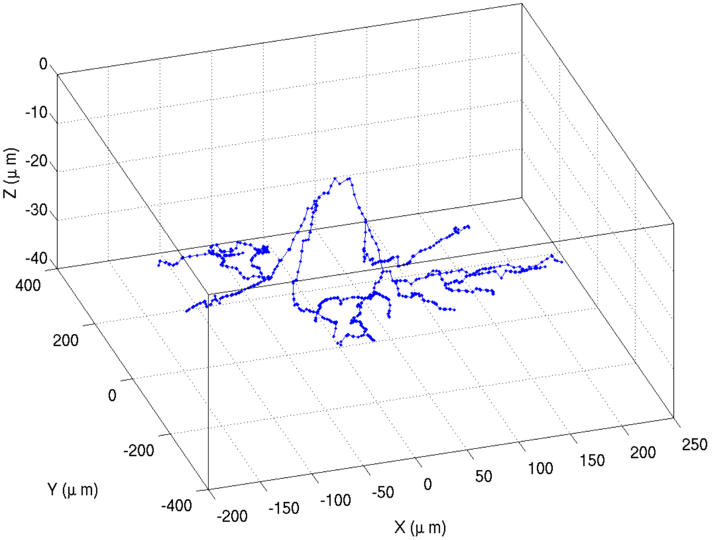
Retinal Ganglion Cells (D. Hubel: *Eye, Brain, & Vision*, '95)



A Typical Neuron (from Wikipedia)



Mouse's RGC as a Graph



The Roadmap

In order to deal with such *Data Analysis on Graphs and Networks*, we need to understand how to *represent* (digital) data and *manipulate* (e.g., compress, filter, . . .) them in general. This requires some basic knowledge on:

- Fourier Analysis, in particular, Fourier series, and its discrete version
- Linear Algebra, in particular, basis vectors, change of bases, linear transformations, eigenvalues and eigenvectors, and singular value decomposition (SVD)
- Graph Theory terminology

For Day 1, we will review the basics of Fourier Analysis and Linear Algebra, mainly, from the viewpoint of *Data Representation and Approximation*. Day 2 will start by reviewing the basics of Graph Theory and then discuss the key tools for Data Representation and Analysis on *Graphs and Networks*.

The Roadmap

In order to deal with such *Data Analysis on Graphs and Networks*, we need to understand how to *represent* (digital) data and *manipulate* (e.g., compress, filter, ...) them in general. This requires some basic knowledge on:

- Fourier Analysis, in particular, Fourier series, and its discrete version
- Linear Algebra, in particular, basis vectors, change of bases, linear transformations, eigenvalues and eigenvectors, and singular value decomposition (SVD)
- Graph Theory terminology

For Day 1, we will review the basics of Fourier Analysis and Linear Algebra, mainly, from the viewpoint of *Data Representation and Approximation*. Day 2 will start by reviewing the basics of Graph Theory and then discuss the key tools for Data Representation and Analysis on *Graphs and Networks*.

The Roadmap

In order to deal with such *Data Analysis on Graphs and Networks*, we need to understand how to *represent* (digital) data and *manipulate* (e.g., compress, filter, ...) them in general. This requires some basic knowledge on:

- Fourier Analysis, in particular, Fourier series, and its discrete version
- Linear Algebra, in particular, basis vectors, change of bases, linear transformations, eigenvalues and eigenvectors, and singular value decomposition (SVD)
- Graph Theory terminology

For Day 1, we will review the basics of Fourier Analysis and Linear Algebra, mainly, from the viewpoint of *Data Representation and Approximation*. Day 2 will start by reviewing the basics of Graph Theory and then discuss the key tools for Data Representation and Analysis on *Graphs and Networks*.

The Roadmap

In order to deal with such *Data Analysis on Graphs and Networks*, we need to understand how to *represent* (digital) data and *manipulate* (e.g., compress, filter, . . .) them in general. This requires some basic knowledge on:

- Fourier Analysis, in particular, Fourier series, and its discrete version
- Linear Algebra, in particular, basis vectors, change of bases, linear transformations, eigenvalues and eigenvectors, and singular value decomposition (SVD)
- Graph Theory terminology

For Day 1, we will review the basics of Fourier Analysis and Linear Algebra, mainly, from the viewpoint of *Data Representation and Approximation*. Day 2 will start by reviewing the basics of Graph Theory and then discuss the key tools for Data Representation and Analysis on *Graphs and Networks*.

The Roadmap

In order to deal with such *Data Analysis on Graphs and Networks*, we need to understand how to *represent* (digital) data and *manipulate* (e.g., compress, filter, ...) them in general. This requires some basic knowledge on:

- Fourier Analysis, in particular, Fourier series, and its discrete version
- Linear Algebra, in particular, basis vectors, change of bases, linear transformations, eigenvalues and eigenvectors, and singular value decomposition (SVD)
- Graph Theory terminology

For Day 1, we will review the basics of Fourier Analysis and Linear Algebra, mainly, from the viewpoint of *Data Representation and Approximation*.

Day 2 will start by reviewing the basics of Graph Theory and then discuss the key tools for Data Representation and Analysis on *Graphs and Networks*.

The Roadmap

In order to deal with such *Data Analysis on Graphs and Networks*, we need to understand how to *represent* (digital) data and *manipulate* (e.g., compress, filter, . . .) them in general. This requires some basic knowledge on:

- Fourier Analysis, in particular, Fourier series, and its discrete version
- Linear Algebra, in particular, basis vectors, change of bases, linear transformations, eigenvalues and eigenvectors, and singular value decomposition (SVD)
- Graph Theory terminology

For Day 1, we will review the basics of Fourier Analysis and Linear Algebra, mainly, from the viewpoint of *Data Representation and Approximation*. Day 2 will start by reviewing the basics of Graph Theory and then discuss the key tools for Data Representation and Analysis on *Graphs and Networks*.

Outline

The 1D Wave Equation

Around mid 18 C, d'Alembert, Euler, D. Bernoulli examined and created the theory behind vibrations of a 1D string.

- Consider a perfectly elastic and flexible string of length ℓ.
- $\rho(x)$: a mass density; $T(x)$: the tension of the string at $x \in [0, \ell]$.
- If $u(x,t)$ is the vertical displacement of the string at location $x \in [0, \ell]$ and time $t \geq 0$, then the string vibrates according to the 1D wave equation (a.k.a. the string equation): $\rho(x)\dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial}{\partial x}\left(T(x)\dfrac{\partial u}{\partial x}\right)$

The 1D Wave Equation

Around mid 18 C, d'Alembert, Euler, D. Bernoulli examined and created the theory behind vibrations of a 1D string.

- Consider a perfectly elastic and flexible string of length ℓ.
- $\rho(x)$: a mass density; $T(x)$: the tension of the string at $x \in [0, \ell]$.
- If $u(x,t)$ is the vertical displacement of the string at location $x \in [0,\ell]$ and time $t \geq 0$, then the string vibrates according to the 1D wave equation (a.k.a. the string equation): $\rho(x)\dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial}{\partial x}\left(T(x)\dfrac{\partial u}{\partial x}\right)$

The 1D Wave Equation

Around mid 18 C, d'Alembert, Euler, D. Bernoulli examined and created the theory behind vibrations of a 1D string.

- Consider a perfectly elastic and flexible string of length ℓ.
- $\rho(x)$: a mass density; $T(x)$: the tension of the string at $x \in [0, \ell]$.
- If $u(x,t)$ is the vertical displacement of the string at location $x \in [0, \ell]$ and time $t \geq 0$, then the string vibrates according to the 1D wave equation (a.k.a. the string equation): $\rho(x) \frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(T(x) \frac{\partial u}{\partial x}\right)$

The 1D Wave Equation

Around mid 18 C, d'Alembert, Euler, D. Bernoulli examined and created the theory behind vibrations of a 1D string.

- Consider a perfectly elastic and flexible string of length ℓ.
- $\rho(x)$: a mass density; $T(x)$: the tension of the string at $x \in [0, \ell]$.
- If $u(x,t)$ is the vertical displacement of the string at location $x \in [0, \ell]$ and time $t \geq 0$, then the string vibrates according to the 1D wave equation (a.k.a. the string equation): $\rho(x) \dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial}{\partial x}\left(T(x) \dfrac{\partial u}{\partial x} \right)$

(a) Jean d'Alembert (1717–1783)

(b) Leonhard Euler (1707–1783)

(c) Daniel Bernoulli (1700–1782)

Importance of the Boundary and Initial Conditions

- From now on, for simplicity, we assume the uniform density and constant tension, i.e., $\rho(x) \equiv \rho$, $T(x) \equiv T$.
- Under this assumption, the above wave equation simplifies to:

$$u_{tt} = c^2 u_{xx} \quad c \equiv \sqrt{T/\rho}.$$

- The 1D wave equation above has infinitely many solutions.
- Need to specify a boundary condition (BC) and an initial condition (IC) to obtain the desired solution.
- One possibility: both ends of the string are held fixed all the time $\Longrightarrow$ the Dirichlet BC: $u(0,t) = u(\ell,t) = 0$, $\forall t \geq 0$.
- As for the IC, let $u(x,0) = f(x)$ (initial position); $u_t(x,0) = g(x)$ (initial velocity), $\forall x \in [0,\ell]$. What we have then is:

$$\begin{cases} u_{tt} = c^2 u_{xx} & \text{for } x \in (0,\ell) \text{ and } t > 0; \\ u(0,t) = u(\ell,t) = 0 & \text{for } t \geq 0; \\ u(x,0) = f(x), \quad u_t(x,0) = g(x) & \text{for } x \in [0,\ell]. \end{cases} \tag{1}$$

Importance of the Boundary and Initial Conditions

- From now on, for simplicity, we assume the uniform density and constant tension, i.e., $\rho(x) \equiv \rho$, $T(x) \equiv T$.
- Under this assumption, the above wave equation simplifies to:

$$u_{tt} = c^2 u_{xx} \quad c \equiv \sqrt{T/\rho}.$$

- The 1D wave equation above has infinitely many solutions.
- Need to specify a boundary condition (BC) and an initial condition (IC) to obtain the desired solution.
- One possibility: both ends of the string are held fixed all the time $\Longrightarrow$ the Dirichlet BC: $u(0,t) = u(\ell,t) = 0, \ \forall t \ge 0$.
- As for the IC, let $u(x,0) = f(x)$ (initial position); $u_t(x,0) = g(x)$ (initial velocity), $\forall x \in [0,\ell]$. What we have then is:

$$\begin{cases} u_{tt} = c^2 u_{xx} & \text{for } x \in (0,\ell) \text{ and } t > 0; \\ u(0,t) = u(\ell,t) = 0 & \text{for } t \ge 0; \\ u(x,0) = f(x), \quad u_t(x,0) = g(x) & \text{for } x \in [0,\ell]. \end{cases} \tag{1}$$

Importance of the Boundary and Initial Conditions

- From now on, for simplicity, we assume the uniform density and constant tension, i.e., $\rho(x) \equiv \rho$, $T(x) \equiv T$.
- Under this assumption, the above wave equation simplifies to:

$$u_{tt} = c^2 u_{xx} \quad c \equiv \sqrt{T/\rho}.$$

- The 1D wave equation above has infinitely many solutions.
- Need to specify a boundary condition (BC) and an initial condition (IC) to obtain the desired solution.
- One possibility: both ends of the string are held fixed all the time $\Longrightarrow$ the Dirichlet BC: $u(0,t) = u(\ell,t) = 0$, $\forall t \geq 0$.
- As for the IC, let $u(x,0) = f(x)$ (initial position); $u_t(x,0) = g(x)$ (initial velocity), $\forall x \in [0,\ell]$. What we have then is:

$$\begin{cases} u_{tt} = c^2 u_{xx} & \text{for } x \in (0,\ell) \text{ and } t > 0; \\ u(0,t) = u(\ell,t) = 0 & \text{for } t \geq 0; \\ u(x,0) = f(x), \quad u_t(x,0) = g(x) & \text{for } x \in [0,\ell]. \end{cases} \tag{1}$$

Importance of the Boundary and Initial Conditions

- From now on, for simplicity, we assume the uniform density and constant tension, i.e., $\rho(x) \equiv \rho$, $T(x) \equiv T$.
- Under this assumption, the above wave equation simplifies to:

$$u_{tt} = c^2 u_{xx} \quad c \equiv \sqrt{T/\rho}.$$

- The 1D wave equation above has infinitely many solutions.
- Need to specify a boundary condition (BC) and an initial condition (IC) to obtain the desired solution.
- One possibility: both ends of the string are held fixed all the time $\Longrightarrow$ the Dirichlet BC: $u(0,t) = u(\ell,t) = 0$, $\forall t \geq 0$.
- As for the IC, let $u(x,0) = f(x)$ (initial position); $u_t(x,0) = g(x)$ (initial velocity), $\forall x \in [0,\ell]$. What we have then is:

$$\begin{cases} u_{tt} = c^2 u_{xx} & \text{for } x \in (0,\ell) \text{ and } t > 0; \\ u(0,t) = u(\ell,t) = 0 & \text{for } t \geq 0; \\ u(x,0) = f(x), \quad u_t(x,0) = g(x) & \text{for } x \in [0,\ell]. \end{cases} \tag{1}$$

Importance of the Boundary and Initial Conditions

- From now on, for simplicity, we assume the uniform density and constant tension, i.e., $\rho(x) \equiv \rho$, $T(x) \equiv T$.
- Under this assumption, the above wave equation simplifies to:

$$u_{tt} = c^2 u_{xx} \quad c \equiv \sqrt{T/\rho}.$$

- The 1D wave equation above has infinitely many solutions.
- Need to specify a boundary condition (BC) and an initial condition (IC) to obtain the desired solution.
- One possibility: both ends of the string are held fixed all the time $\Longrightarrow$ the Dirichlet BC: $u(0,t) = u(\ell,t) = 0$, $\forall t \ge 0$.
- As for the IC, let $u(x,0) = f(x)$ (initial position); $u_t(x,0) = g(x)$ (initial velocity), $\forall x \in [0,\ell]$. What we have then is:

$$\begin{cases} u_{tt} = c^2 u_{xx} & \text{for } x \in (0,\ell) \text{ and } t > 0; \\ u(0,t) = u(\ell,t) = 0 & \text{for } t \ge 0; \\ u(x,0) = f(x), \quad u_t(x,0) = g(x) & \text{for } x \in [0,\ell]. \end{cases} \tag{1}$$

Importance of the Boundary and Initial Conditions

- From now on, for simplicity, we assume the uniform density and constant tension, i.e., $\rho(x) \equiv \rho$, $T(x) \equiv T$.
- Under this assumption, the above wave equation simplifies to:

$$u_{tt} = c^2 u_{xx} \quad c \equiv \sqrt{T/\rho}.$$

- The 1D wave equation above has infinitely many solutions.
- Need to specify a boundary condition (BC) and an initial condition (IC) to obtain the desired solution.
- One possibility: both ends of the string are held fixed all the time $\Longrightarrow$ the Dirichlet BC: $u(0,t) = u(\ell,t) = 0$, $\forall t \geq 0$.
- As for the IC, let $u(x,0) = f(x)$ (initial position); $u_t(x,0) = g(x)$ (initial velocity), $\forall x \in [0,\ell]$. What we have then is:

$$\begin{cases} u_{tt} = c^2 u_{xx} & \text{for } x \in (0,\ell) \text{ and } t > 0; \\ u(0,t) = u(\ell,t) = 0 & \text{for } t \geq 0; \\ u(x,0) = f(x), \quad u_t(x,0) = g(x) & \text{for } x \in [0,\ell]. \end{cases} \tag{1}$$

Various Boundary Conditions

- The Dirichlet BC: both ends are *fixed*:

$$u(0,t) = u(\ell,t) = 0.$$

- The Neumann BC: both ends are *free* to move transversally:

$$u_x(0,t) = u_x(\ell,t) = 0.$$

- The Robin (a.k.a. impedance) BC: a *linearly restorative transverse force* is applied at both ends:

$$a_0 u(0,t) + b_0 u_x(0,t) = a_\ell u(\ell,t) + b_\ell u_x(\ell,t) = 0, \quad a_i \neq 0 \neq b_i, i = 0, \ell.$$

(a) J.P.G.L. Dirichlet (1805–1859)

Various Boundary Conditions

- The Dirichlet BC: both ends are *fixed*:

$$u(0,t) = u(\ell,t) = 0.$$

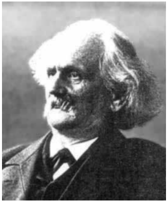
- The Neumann BC: both ends are *free* to move transversally:

$$u_x(0,t) = u_x(\ell,t) = 0.$$

- The Robin (a.k.a. impedance) BC: a *linearly restorative transverse force* is applied at both ends:

$$a_0 u(0,t) + b_0 u_x(0,t) = a_\ell u(\ell,t) + b_\ell u_x(\ell,t) = 0, \quad a_i \neq 0 \neq b_i, i = 0, \ell.$$

(a) J.P.G.L. Dirichlet (1805–1859)



(b) Carl Neumann (1832–1925)

Various Boundary Conditions

- The Dirichlet BC: both ends are *fixed*:

$$u(0,t) = u(\ell,t) = 0.$$

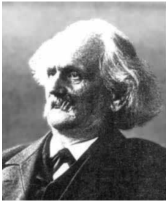
- The Neumann BC: both ends are *free* to move transversally:

$$u_x(0,t) = u_x(\ell,t) = 0.$$

- The Robin (a.k.a. impedance) BC: a *linearly restorative transverse force* is applied at both ends:

$$a_0 u(0,t) + b_0 u_x(0,t) = a_\ell u(\ell,t) + b_\ell u_x(\ell,t) = 0, \quad a_i \neq 0 \neq b_i, i = 0, \ell.$$

(a) J.P.G.L. Dirichlet (1805–1859)



(b) Carl Neumann (1832–1925)

(c) Gustave Robin (1855–1897)

Behavior of the String $u(x,t)$

- Use the method of separation of variables to seek a nontrivial solution of the form: $u(x,t) = X(x)T(t)$.
- Plugging $X(x)T(t)$ into the (1), we get:

$$XT'' = c^2 X'' T \Longrightarrow \frac{X''}{X} = \frac{T''}{c^2 T} = k,$$

  where k must be a *constant*.
- This leads to the following ODEs:

$$X'' - kX = 0 \quad \text{with } X(0) = X(\ell) = 0, \tag{2}$$

$$T'' - c^2 kT = 0 \tag{3}$$

- Eqn. (2) is the simplest example of the Laplacian eigenvalue problem.
- The characteristic equation of (2), i.e., $r^2 - k = 0$, must be analyzed carefully.

Behavior of the String $u(x, t)$

- Use the method of separation of variables to seek a nontrivial solution of the form: $u(x, t) = X(x)T(t)$.
- Plugging $X(x)T(t)$ into the (1), we get:

$$XT'' = c^2 X''T \Longrightarrow \frac{X''}{X} = \frac{T''}{c^2 T} = k,$$

where k must be a *constant*.

- This leads to the following ODEs:

$$X'' - kX = 0 \quad \text{with } X(0) = X(\ell) = 0, \tag{2}$$

$$T'' - c^2 kT = 0 \tag{3}$$

- Eqn. (2) is the simplest example of the Laplacian eigenvalue problem.
- The characteristic equation of (2), i.e., $r^2 - k = 0$, must be analyzed carefully.

Behavior of the String $u(x,t)$

- Use the method of separation of variables to seek a nontrivial solution of the form: $u(x,t) = X(x)T(t)$.
- Plugging $X(x)T(t)$ into the (1), we get:

$$XT'' = c^2 X'' T \Longrightarrow \frac{X''}{X} = \frac{T''}{c^2 T} = k,$$

  where k must be a *constant*.
- This leads to the following ODEs:

$$X'' - kX = 0 \quad \text{with } X(0) = X(\ell) = 0, \tag{2}$$

$$T'' - c^2 kT = 0 \tag{3}$$

- Eqn. (2) is the simplest example of the Laplacian eigenvalue problem.
- The characteristic equation of (2), i.e., $r^2 - k = 0$, must be analyzed carefully.

Behavior of the String $u(x,t)$

- Use the method of separation of variables to seek a nontrivial solution of the form: $u(x,t) = X(x)T(t)$.
- Plugging $X(x)T(t)$ into the (1), we get:

$$XT'' = c^2 X'' T \Longrightarrow \frac{X''}{X} = \frac{T''}{c^2 T} = k,$$

where k must be a *constant*.
- This leads to the following ODEs:

$$X'' - kX = 0 \quad \text{with } X(0) = X(\ell) = 0, \tag{2}$$

$$T'' - c^2 kT = 0 \tag{3}$$

- Eqn. (2) is the simplest example of the Laplacian eigenvalue problem.
- The characteristic equation of (2), i.e., $r^2 - k = 0$, must be analyzed carefully.

Behavior of the String $u(x,t)$

- Use the method of separation of variables to seek a nontrivial solution of the form: $u(x,t) = X(x)T(t)$.
- Plugging $X(x)T(t)$ into the (1), we get:

$$XT'' = c^2 X'' T \Longrightarrow \frac{X''}{X} = \frac{T''}{c^2 T} = k,$$

  where k must be a *constant*.
- This leads to the following ODEs:

$$X'' - kX = 0 \quad \text{with } X(0) = X(\ell) = 0, \tag{2}$$

$$T'' - c^2 kT = 0 \tag{3}$$

- Eqn. (2) is the simplest example of the Laplacian eigenvalue problem.
- The characteristic equation of (2), i.e., $r^2 - k = 0$, must be analyzed carefully.

Solving ODEs

Case I: $k > 0 \Longrightarrow r = \pm\sqrt{k}$; hence

$$X(x) = A\mathrm{e}^{\sqrt{k}x} + B\mathrm{e}^{-\sqrt{k}x} \quad \text{or} \quad A\cosh(\sqrt{k}x) + B\sinh(\sqrt{k}x).$$

Applying the BC $X(0) = X(\ell) = 0$ yields $A = B = 0$, thus the case of $k > 0$ is *not feasible*.

Case II: $k = 0 \Longrightarrow X'' = 0 \Longrightarrow X(x) = Ax + B$, which again leads to $X(x) \equiv 0$.

Case III: $k < 0$. Set $k = -\xi^2$ and $\xi > 0$. Then the characteristic equation becomes $r^2 + \xi^2 = 0$, i.e., $r = \pm\mathrm{i}\xi$. Therefore we get

$$X(x) = A\cos(\xi x) + B\sin(\xi x)$$

By the BC $X(0) = X(\ell) = 0$, we get:

$$\begin{cases} X(0) = 0 & \Longrightarrow \quad A = 0 \\ X(\ell) = B\sin(\xi\ell) = 0 & \Longrightarrow \quad \xi = \frac{n\pi}{\ell}, \quad \forall n \in \mathbb{N} \end{cases}$$

Note $n = 0$ leads to $X(x) \equiv 0$ in this case, so it should not be included.

Solving ODEs

Case I: $k > 0 \Longrightarrow r = \pm\sqrt{k}$; hence

$$X(x) = Ae^{\sqrt{k}x} + Be^{-\sqrt{k}x} \quad \text{or} \quad A\cosh(\sqrt{k}x) + B\sinh(\sqrt{k}x).$$

Applying the BC $X(0) = X(\ell) = 0$ yields $A = B = 0$, thus the case of $k > 0$ is *not feasible*.

Case II: $k = 0 \Longrightarrow X'' = 0 \Longrightarrow X(x) = Ax + B$, which again leads to $X(x) \equiv 0$.

Case III: $k < 0$. Set $k = -\xi^2$ and $\xi > 0$. Then the characteristic equation becomes $r^2 + \xi^2 = 0$, i.e., $r = \pm i\xi$. Therefore we get

$$X(x) = A\cos(\xi x) + B\sin(\xi x)$$

By the BC $X(0) = X(\ell) = 0$, we get:

$$\begin{cases} X(0) = 0 & \Longrightarrow \quad A = 0 \\ X(\ell) = B\sin(\xi\ell) = 0 & \Longrightarrow \quad \xi = \frac{n\pi}{\ell}, \quad \forall n \in \mathbb{N} \end{cases}$$

Note $n = 0$ leads to $X(x) \equiv 0$ in this case, so it should not be included.

Solving ODEs

Case I: $k > 0 \Longrightarrow r = \pm\sqrt{k}$; hence

$$X(x) = Ae^{\sqrt{k}x} + Be^{-\sqrt{k}x} \quad \text{or} \quad A\cosh(\sqrt{k}x) + B\sinh(\sqrt{k}x).$$

Applying the BC $X(0) = X(\ell) = 0$ yields $A = B = 0$, thus the case of $k > 0$ is *not feasible*.

Case II: $k = 0 \Longrightarrow X'' = 0 \Longrightarrow X(x) = Ax + B$, which again leads to $X(x) \equiv 0$.

Case III: $k < 0$. Set $k = -\xi^2$ and $\xi > 0$. Then the characteristic equation becomes $r^2 + \xi^2 = 0$, i.e., $r = \pm i\xi$. Therefore we get

$$X(x) = A\cos(\xi x) + B\sin(\xi x)$$

By the BC $X(0) = X(\ell) = 0$, we get:

$$\begin{cases} X(0) = 0 & \Longrightarrow \quad A = 0 \\ X(\ell) = B\sin(\xi\ell) = 0 & \Longrightarrow \quad \xi = \frac{n\pi}{\ell}, \quad \forall n \in \mathbb{N} \end{cases}$$

Note $n = 0$ leads to $X(x) \equiv 0$ in this case, so it should not be included.

Forming the Solution

- Hence we have $X(x) = B\sin(\frac{n\pi}{\ell}x)$, and for convenience, by setting $B = \sqrt{\frac{2}{\ell}}$, let us define

$$X_n(x) = \varphi_n(x) := \sqrt{\frac{2}{\ell}}\sin\left(\frac{n\pi}{\ell}x\right).$$

- Similarly, by $T'' = -\xi^2 c^2 T$ we obtain the family of solutions

$$T_n(t) = a_n\cos\left(\frac{n\pi c}{\ell}t\right) + b_n\sin\left(\frac{n\pi c}{\ell}t\right).$$

- Now, for each $n \in \mathbb{N}$, the function

$$u_n(x,t) = T_n(t)\cdot\varphi_n(x) = \left\{a_n\cos\left(\frac{n\pi c}{\ell}t\right) + b_n\sin\left(\frac{n\pi c}{\ell}t\right)\right\}\sqrt{\frac{2}{\ell}}\sin\left(\frac{n\pi}{\ell}x\right)$$

satisfies (1).

Forming the Solution

- Hence we have $X(x) = B\sin(\frac{n\pi}{\ell}x)$, and for convenience, by setting $B = \sqrt{\frac{2}{\ell}}$, let us define

$$X_n(x) = \varphi_n(x) := \sqrt{\frac{2}{\ell}}\sin\left(\frac{n\pi}{\ell}x\right).$$

- Similarly, by $T'' = -\xi^2 c^2 T$ we obtain the family of solutions

$$T_n(t) = a_n\cos\left(\frac{n\pi c}{\ell}t\right) + b_n\sin\left(\frac{n\pi c}{\ell}t\right).$$

- Now, for each $n \in \mathbb{N}$, the function

$$u_n(x,t) = T_n(t)\cdot\varphi_n(x) = \left\{a_n\cos\left(\frac{n\pi c}{\ell}t\right) + b_n\sin\left(\frac{n\pi c}{\ell}t\right)\right\}\sqrt{\frac{2}{\ell}}\sin\left(\frac{n\pi}{\ell}x\right)$$

satisfies (1).

Forming the Solution

- Hence we have $X(x) = B\sin(\frac{n\pi}{\ell}x)$, and for convenience, by setting $B = \sqrt{\frac{2}{\ell}}$, let us define

$$X_n(x) = \varphi_n(x) := \sqrt{\frac{2}{\ell}}\sin\left(\frac{n\pi}{\ell}x\right).$$

- Similarly, by $T'' = -\xi^2 c^2 T$ we obtain the family of solutions

$$T_n(t) = a_n\cos\left(\frac{n\pi c}{\ell}t\right) + b_n\sin\left(\frac{n\pi c}{\ell}t\right).$$

- Now, for each $n \in \mathbb{N}$, the function

$$u_n(x,t) = T_n(t)\cdot\varphi_n(x) = \left\{a_n\cos\left(\frac{n\pi c}{\ell}t\right) + b_n\sin\left(\frac{n\pi c}{\ell}t\right)\right\}\sqrt{\frac{2}{\ell}}\sin\left(\frac{n\pi}{\ell}x\right)$$

satisfies (1).

Forming the Solution ...

- Hence, by the *Superposition Principle*,

$$u(x,t) = \sum_{n=1}^{\infty} u_n(x,t) = \sum_{n=1}^{\infty} \left\{ a_n \cos\left(\frac{n\pi c}{\ell} t\right) + b_n \sin\left(\frac{n\pi c}{\ell} t\right) \right\} \varphi_n(x) \quad (4)$$

is a general solution with yet undetermined coefficients a_n and b_n.

- Next, we specify the coefficients a_n and b_n by matching (4) with the ICs in (1). Thus we get

$$u(x,0) = f(x) = \sum_{n=1}^{\infty} a_n \varphi_n(x) = \sum_{n=1}^{\infty} a_n \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right)$$

- How can we compute $\{a_n\}, \{b_n\}$?

Forming the Solution ...

- Hence, by the *Superposition Principle*,

$$u(x,t) = \sum_{n=1}^{\infty} u_n(x,t) = \sum_{n=1}^{\infty} \left\{ a_n \cos\left(\frac{n\pi c}{\ell} t\right) + b_n \sin\left(\frac{n\pi c}{\ell} t\right) \right\} \varphi_n(x) \quad (4)$$

  is a general solution with yet undetermined coefficients a_n and b_n.

- Next, we specify the coefficients a_n and b_n by matching (4) with the ICs in (1). Thus we get

$$u(x,0) = f(x) = \sum_{n=1}^{\infty} a_n \varphi_n(x) = \sum_{n=1}^{\infty} a_n \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right)$$

- How can we compute $\{a_n\}, \{b_n\}$?

Forming the Solution ...

- Hence, by the *Superposition Principle*,

$$u(x,t) = \sum_{n=1}^{\infty} u_n(x,t) = \sum_{n=1}^{\infty} \left\{ a_n \cos\left(\frac{n\pi c}{\ell} t\right) + b_n \sin\left(\frac{n\pi c}{\ell} t\right) \right\} \varphi_n(x) \quad (4)$$

  is a general solution with yet undetermined coefficients a_n and b_n.
- Next, we specify the coefficients a_n and b_n by matching (4) with the ICs in (1). Thus we get

$$u(x,0) = f(x) = \sum_{n=1}^{\infty} a_n \varphi_n(x) = \sum_{n=1}^{\infty} a_n \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right)$$

- How can we compute $\{a_n\}, \{b_n\}$?

An Orthonormal Basis

- It turns out that $\left\{\varphi_n(x) = \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right)\right\}_{n\in\mathbb{N}}$ form an orthonormal basis of *the space of square integrable (= finite energy) functions* denoted by $L^2[0,\ell]$.
- In other words, every finite energy function $f(x)$ defined on $[0,\ell]$ can be written as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$, i.e.,

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Note that this linear combination may have infinite terms, and $L^2[0,\ell]$ is an example of the so-called (infinite dimensional) Hilbert space where the notion of the inner product can be defined: $\langle f, g\rangle := \int_0^\ell f(x)\, g(x)\, \mathrm{d}x$.
- Then, you can show easily that $\{\varphi_n\}_{n\in\mathbb{N}}$ form an orthonormal set (an exercise!):

$$\langle \varphi_n, \varphi_{n'}\rangle = \delta_{nn'} := \begin{cases} 1 & \text{if } n = n'; \\ 0 & \text{otherwise,} \end{cases}$$

where $\delta_{n,n'}$ is called *Kronecker's delta*.
- In an infinite dimensional Hilbert space, a general orthonormal set does not necessarily form a basis. However, one can show that $\{\varphi_n\}_{k\in\mathbb{N}}$ forms a *complete* orthonormal set (= an orthonormal basis) of $L^2[0,\ell]$.

An Orthonormal Basis

- It turns out that $\left\{ \varphi_n(x) = \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right) \right\}_{n \in \mathbb{N}}$ form an orthonormal basis of *the space of square integrable (= finite energy) functions* denoted by $L^2[0, \ell]$.
- In other words, every finite energy function $f(x)$ defined on $[0, \ell]$ can be written as a linear combination of $\{\varphi_n(x)\}_{n \in \mathbb{N}}$, i.e.,

$$f(x) \sim a_1 \varphi_1(x) + \cdots + a_n \varphi_n(x) + \cdots$$

- Note that this linear combination may have infinite terms, and $L^2[0, \ell]$ is an example of the so-called (infinite dimensional) Hilbert space where the notion of the inner product can be defined: $\langle f, g \rangle := \int_0^\ell f(x)\, g(x)\, \mathrm{d}x$.
- Then, you can show easily that $\{\varphi_n\}_{n \in \mathbb{N}}$ form an orthonormal set (an exercise!):

$$\langle \varphi_n, \varphi_{n'} \rangle = \delta_{nn'} := \begin{cases} 1 & \text{if } n = n'; \\ 0 & \text{otherwise,} \end{cases}$$

where $\delta_{n,n'}$ is called *Kronecker's delta*.

- In an infinite dimensional Hilbert space, a general orthonormal set does not necessarily form a basis. However, one can show that $\{\varphi_n\}_{k \in \mathbb{N}}$ forms a *complete* orthonormal set (= an orthonormal basis) of $L^2[0, \ell]$.

An Orthonormal Basis

- It turns out that $\left\{\varphi_n(x) = \sqrt{\frac{2}{\ell}}\sin\left(\frac{n\pi}{\ell}x\right)\right\}_{n\in\mathbb{N}}$ form an orthonormal basis of *the space of square integrable (= finite energy) functions* denoted by $L^2[0,\ell]$.
- In other words, every finite energy function $f(x)$ defined on $[0,\ell]$ can be written as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$, i.e.,

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Note that this linear combination may have infinite terms, and $L^2[0,\ell]$ is an example of the so-called (infinite dimensional) Hilbert space where the notion of the inner product can be defined: $\langle f, g\rangle := \int_0^\ell f(x)\,g(x)\,\mathrm{d}x$.
- Then, you can show easily that $\{\varphi_n\}_{n\in\mathbb{N}}$ form an orthonormal set (an exercise!):

$$\langle \varphi_n, \varphi_{n'}\rangle = \delta_{nn'} := \begin{cases} 1 & \text{if } n = n'; \\ 0 & \text{otherwise,} \end{cases}$$

  where $\delta_{n,n'}$ is called *Kronecker's delta*.
- In an infinite dimensional Hilbert space, a general orthonormal set does not necessarily form a basis. However, one can show that $\{\varphi_n\}_{k\in\mathbb{N}}$ forms a *complete* orthonormal set (= an orthonormal basis) of $L^2[0,\ell]$.

An Orthonormal Basis

- It turns out that $\left\{\varphi_n(x) = \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right)\right\}_{n\in\mathbb{N}}$ form an orthonormal basis of *the space of square integrable (= finite energy) functions* denoted by $L^2[0,\ell]$.
- In other words, every finite energy function $f(x)$ defined on $[0,\ell]$ can be written as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$, i.e.,

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Note that this linear combination may have infinite terms, and $L^2[0,\ell]$ is an example of the so-called (infinite dimensional) Hilbert space where the notion of the inner product can be defined: $\langle f, g\rangle := \int_0^\ell f(x)\, g(x)\, \mathrm{d}x$.
- Then, you can show easily that $\{\varphi_n\}_{n\in\mathbb{N}}$ form an orthonormal set (an exercise!):

$$\langle \varphi_n, \varphi_{n'}\rangle = \delta_{nn'} := \begin{cases} 1 & \text{if } n = n'; \\ 0 & \text{otherwise,} \end{cases}$$

  where $\delta_{n,n'}$ is called *Kronecker's delta*.
- In an infinite dimensional Hilbert space, a general orthonormal set does not necessarily form a basis. However, one can show that $\{\varphi_n\}_{k\in\mathbb{N}}$ forms a *complete* orthonormal set (= an orthonormal basis) of $L^2[0,\ell]$.

An Orthonormal Basis

- It turns out that $\left\{\varphi_n(x) = \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right)\right\}_{n\in\mathbb{N}}$ form an orthonormal basis of *the space of square integrable (= finite energy) functions* denoted by $L^2[0,\ell]$.
- In other words, every finite energy function $f(x)$ defined on $[0,\ell]$ can be written as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$, i.e.,

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Note that this linear combination may have infinite terms, and $L^2[0,\ell]$ is an example of the so-called (infinite dimensional) Hilbert space where the notion of the inner product can be defined: $\langle f, g\rangle := \int_0^\ell f(x)\, g(x)\, \mathrm{d}x$.
- Then, you can show easily that $\{\varphi_n\}_{n\in\mathbb{N}}$ form an orthonormal set (an exercise!):

$$\langle \varphi_n, \varphi_{n'}\rangle = \delta_{nn'} := \begin{cases} 1 & \text{if } n = n'; \\ 0 & \text{otherwise,} \end{cases}$$

  where $\delta_{n,n'}$ is called *Kronecker's delta*.
- In an infinite dimensional Hilbert space, a general orthonormal set does not necessarily form a basis. However, one can show that $\{\varphi_n\}_{k\in\mathbb{N}}$ forms a *complete* orthonormal set (= an orthonormal basis) of $L^2[0,\ell]$.

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0, \ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis ...

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

An Orthonormal Basis . . .

- Why such an orthonormal basis is important?
- Because it allows us to write any $f \in L^2[0,\ell]$ as a linear combination of $\{\varphi_n(x)\}_{n\in\mathbb{N}}$:

$$f(x) \sim a_1\varphi_1(x) + \cdots + a_n\varphi_n(x) + \cdots$$

- Moreover, computing the coefficients $\{a_n\}$ is relatively easy: take the inner product of both sides with f with φ_k, (i.e., multiply φ_k to the both sides and integrate it) gives us:

$$\begin{aligned}
\langle f, \varphi_k \rangle &= \langle a_1\varphi_1 + \cdots + a_k\varphi_k + \cdots + a_n\varphi_n + \cdots, \varphi_k \rangle \\
&= \langle a_1\varphi_1, \varphi_k \rangle + \cdots + \langle a_k\varphi_k, \varphi_k \rangle + \cdots + \langle a_n\varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \langle \varphi_1, \varphi_k \rangle + \cdots + a_k \langle \varphi_k, \varphi_k \rangle + \cdots + a_n \langle \varphi_n, \varphi_k \rangle + \cdots \\
&= a_1 \cdot \delta_{1,k} + \cdots + a_k \cdot \delta_{k,k} + \cdots + a_n \cdot \delta_{n,k} + \cdots \\
&= a_1 \cdot 0 + \cdots + a_k \cdot 1 + \cdots + a_n \cdot 0 + \cdots \\
&= a_k
\end{aligned}$$

Forming the Solution . . .

- Similarly, $u_t(x,0) = g(x) = \sum_{n=1}^{\infty} \frac{n\pi c}{\ell} b_n \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell}x\right)$.
- Note that $\frac{n\pi c}{\ell} b_n = \langle g, \varphi_n \rangle \Longrightarrow b_n = \frac{\ell}{n\pi c}\langle g, \varphi_n \rangle$.
- Finally, we obtain the particular solution:

$$u(x,t) = \sum_{n=1}^{\infty} \left\{ \langle f, \varphi_n \rangle \cos\left(\frac{n\pi c}{\ell}t\right) + \frac{\ell}{n\pi c}\langle g, \varphi_n \rangle \sin\left(\frac{n\pi c}{\ell}t\right) \right\} \varphi_n(x),$$

which satisfies (1) completely including both BC & IC.

Figure: Jean Baptiste Joseph Fourier (1768–1830)

Forming the Solution ...

- Similarly, $u_t(x,0) = g(x) = \sum_{n=1}^{\infty} \frac{n\pi c}{\ell} b_n \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell}x\right)$.
- Note that $\frac{n\pi c}{\ell} b_n = \langle g, \varphi_n \rangle \Longrightarrow b_n = \frac{\ell}{n\pi c} \langle g, \varphi_n \rangle$.
- Finally, we obtain the particular solution:

$$u(x,t) = \sum_{n=1}^{\infty} \left\{ \langle f, \varphi_n \rangle \cos\left(\frac{n\pi c}{\ell}t\right) + \frac{\ell}{n\pi c} \langle g, \varphi_n \rangle \sin\left(\frac{n\pi c}{\ell}t\right) \right\} \varphi_n(x),$$

which satisfies (1) completely including both BC & IC.

Figure: Jean Baptiste Joseph Fourier (1768–1830)

Forming the Solution ...

- Similarly, $u_t(x,0) = g(x) = \sum_{n=1}^{\infty} \frac{n\pi c}{\ell} b_n \sqrt{\frac{2}{\ell}} \sin\left(\frac{n\pi}{\ell} x\right)$.
- Note that $\frac{n\pi c}{\ell} b_n = \langle g, \varphi_n \rangle \Longrightarrow b_n = \frac{\ell}{n\pi c} \langle g, \varphi_n \rangle$.
- Finally, we obtain the particular solution:

$$u(x,t) = \sum_{n=1}^{\infty} \left\{ \langle f, \varphi_n \rangle \cos\left(\frac{n\pi c}{\ell} t\right) + \frac{\ell}{n\pi c} \langle g, \varphi_n \rangle \sin\left(\frac{n\pi c}{\ell} t\right) \right\} \varphi_n(x),$$

which satisfies (1) completely including both BC & IC.

Figure: Jean Baptiste Joseph Fourier (1768–1830)

Numerical Simulations

- Using MATLAB®, I simulated the solutions of 1D wave equation under *the Dirichlet BC* with the following parameters: $\ell = 1$(m); $c = 1$(m/s), $g(x) \equiv 0$(m/s).
- Then two initial displacements were considered: $f(x) = \sin^2(\pi x)$ and $f(x) = \mathrm{e}^{-(x-0.5)^2/0.01}$.

Play/Pause | $f(x) = \sin^2(\pi x)$ | $f(x) = \mathrm{e}^{-(x-0.5)^2/0.01}$

Remarks

- Need to check if our solution makes sense physically. Notice that

$$c^2 = \frac{T}{\rho} \Longrightarrow \text{the sound frequency} = \frac{n\pi}{\ell}\sqrt{\frac{T}{\rho}}.$$

- Hence, ℓ is short, T is high, and ρ is small (thin), then such a string generates a high frequency tone.
- On the other hand, if ℓ is long, T is low, and ρ is large (thick), then it generates a low frequency tone.
- Note that the Neumann BC imposes

$$u_x(0,t) = u_x(\ell,t) = 0 \quad \forall t > 0.$$

This leads to the Fourier cosine series expansions of f and g. Note that the Neumann problem allows the solution $u_0(x,t) = a_0 = \text{const}$.

Remarks

- Need to check if our solution makes sense physically. Notice that

$$c^2 = \frac{T}{\rho} \Longrightarrow \text{ the sound frequency } = \frac{n\pi}{\ell}\sqrt{\frac{T}{\rho}}.$$

- Hence, ℓ is short, T is high, and ρ is small (thin), then such a string generates a high frequency tone.
- On the other hand, if ℓ is long, T is low, and ρ is large (thick), then it generates a low frequency tone.
- Note that the Neumann BC imposes

$$u_x(0,t) = u_x(\ell,t) = 0 \quad \forall\, t > 0.$$

This leads to the Fourier cosine series expansions of f and g. Note that the Neumann problem allows the solution $u_0(x,t) = a_0 = \text{const.}$

Remarks

- Need to check if our solution makes sense physically. Notice that

$$c^2 = \frac{T}{\rho} \Longrightarrow \text{ the sound frequency } = \frac{n\pi}{\ell}\sqrt{\frac{T}{\rho}}.$$

- Hence, ℓ is short, T is high, and ρ is small (thin), then such a string generates a high frequency tone.
- On the other hand, if ℓ is long, T is low, and ρ is large (thick), then it generates a low frequency tone.
- Note that the Neumann BC imposes

$$u_x(0,t) = u_x(\ell,t) = 0 \quad \forall t > 0.$$

This leads to the Fourier cosine series expansions of f and g. Note that the Neumann problem allows the solution $u_0(x,t) = a_0 = \text{const.}$

Remarks

- Need to check if our solution makes sense physically. Notice that

$$c^2 = \frac{T}{\rho} \Longrightarrow \text{ the sound frequency } = \frac{n\pi}{\ell}\sqrt{\frac{T}{\rho}}.$$

- Hence, ℓ is short, T is high, and ρ is small (thin), then such a string generates a high frequency tone.
- On the other hand, if ℓ is long, T is low, and ρ is large (thick), then it generates a low frequency tone.
- Note that the Neumann BC imposes

$$u_x(0,t) = u_x(\ell,t) = 0 \quad \forall t > 0.$$

This leads to the Fourier cosine series expansions of f and g. Note that the Neumann problem allows the solution $u_0(x,t) = a_0 = \text{const}$.

Numerical Simulations under the Neumann BC

- Now, let's do the simulation of the 1D wave equation under *the Neumann BC* with the same parameters as before: $\ell = 1(\mathrm{m})$; $c = 1(\mathrm{m/s})$, $g(x) \equiv 0(\mathrm{m/s})$.
- Again the same two initial displacements were considered: $f(x) = \sin^2(\pi x)$ and $f(x) = \mathrm{e}^{-(x-0.5)^2/0.01}$.

Play/Pause | $f(x) = \sin^2(\pi x)$ | $f(x) = \mathrm{e}^{-(x-0.5)^2/0.01}$

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

$$\text{Eigenvalues: } \lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N};$$

$$\text{Eigenfunctions: } \varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D}\, x\right) \quad n \in \mathbb{N}.$$

- In the case of the Neumann-Laplacian, we got

$$\text{Eigenvalues: } \lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N};$$

$$\text{Eigenfunctions: } \varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N}\, x\right) \quad n \in \mathbb{N}_0.$$

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

$$\text{Eigenvalues: } \lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N};$$

$$\text{Eigenfunctions: } \varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D}\, x\right) \quad n \in \mathbb{N}.$$

- In the case of the Neumann-Laplacian, we got

$$\text{Eigenvalues: } \lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N};$$

$$\text{Eigenfunctions: } \varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N}\, x\right) \quad n \in \mathbb{N}_0.$$

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

  Eigenvalues: $\lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N};$

  Eigenfunctions: $\varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D}\, x\right) \quad n \in \mathbb{N}.$

- In the case of the Neumann-Laplacian, we got

  Eigenvalues: $\lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N};$

  Eigenfunctions: $\varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N}\, x\right) \quad n \in \mathbb{N}_0.$

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

  Eigenvalues: $\lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}$;

  Eigenfunctions: $\varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D}\, x\right) \quad n \in \mathbb{N}$.

- In the case of the Neumann-Laplacian, we got

  Eigenvalues: $\lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N}$;

  Eigenfunctions: $\varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N}\, x\right) \quad n \in \mathbb{N}_0$.

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

Eigenvalues: $\lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N};$

Eigenfunctions: $\varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D}\, x\right) \quad n \in \mathbb{N}.$

- In the case of the Neumann-Laplacian, we got

Eigenvalues: $\lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N};$

Eigenfunctions: $\varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N}\, x\right) \quad n \in \mathbb{N}_0.$

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

Eigenvalues: $\lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N};$

Eigenfunctions: $\varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D}\, x\right) \quad n \in \mathbb{N}.$

- In the case of the Neumann-Laplacian, we got

Eigenvalues: $\lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N};$

Eigenfunctions: $\varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N}\, x\right) \quad n \in \mathbb{N}_0.$

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

Eigenvalues: $\lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N};$

Eigenfunctions: $\varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D}\, x\right) \quad n \in \mathbb{N}.$

- In the case of the Neumann-Laplacian, we got

Eigenvalues: $\lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N};$

Eigenfunctions: $\varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N}\, x\right) \quad n \in \mathbb{N}_0.$

Remarks . . .

- Through the separation of variables for finding a solution to the 1D string equation with BC & IC (1), we arrive at the system

$$-X'' = \xi^2 X \quad \text{with } X(0) = X(\ell) = 0. \tag{5}$$

- Notice that (5) is a 1D version of the Dirichlet-Laplacian eigenvalue problem with the domain $\Omega = (0, \ell)$.
- More importantly, we obtained two objects, namely:

Eigenvalues: $\lambda_n^D = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N};$

Eigenfunctions: $\varphi_n^D(x) = \sqrt{\frac{2}{\ell}} \sin\left(\sqrt{\lambda_n^D} x\right) \quad n \in \mathbb{N}.$

- In the case of the Neumann-Laplacian, we got

Eigenvalues: $\lambda_n^N = \left(\frac{n\pi}{\ell}\right)^2 \quad n \in \mathbb{N}_0 := \{0\} \cup \mathbb{N};$

Eigenfunctions: $\varphi_n^N(x) = \sqrt{\frac{2}{\ell}} \cos\left(\sqrt{\lambda_n^N} x\right) \quad n \in \mathbb{N}_0.$

Remarks . . .

- We see that in either BCs, $\{\lambda_n\}_{n=1}^{\infty}$ contains *geometric information* of the domain $\Omega = (0, \ell)$.
- For instance, the size of the first eigenvalue, $\lambda_1 = (\pi/\ell)^2$ tells us the volume of Ω (i.e., the length ℓ of Ω in 1D).
- Under our assumption of constant tension and constant density,

$$\text{small } \lambda_1 \iff \text{long } \ell$$
$$\text{large } \lambda_1 \iff \text{short } \ell$$

- Furthermore, the set $\{\varphi_n\}_{n=1}^{\infty}$ forms an orthonormal basis for $L^2(\Omega)$, so the eigenfunctions allows us to analyze functions living on Ω.
- Next, we will discuss how to represent a signal sampled on a regular lattice (or grids), i.e., a discrete signal instead of a signal in the continuum $[0, \ell]$.

Remarks . . .

- We see that in either BCs, $\{\lambda_n\}_{n=1}^{\infty}$ contains *geometric information* of the domain $\Omega = (0, \ell)$.
- For instance, the size of the first eigenvalue, $\lambda_1 = (\pi/\ell)^2$ tells us the volume of Ω (i.e., the length ℓ of Ω in 1D).
- Under our assumption of constant tension and constant density,

$$\text{small } \lambda_1 \iff \text{long } \ell$$
$$\text{large } \lambda_1 \iff \text{short } \ell$$

- Furthermore, the set $\{\varphi_n\}_{n=1}^{\infty}$ forms an orthonormal basis for $L^2(\Omega)$, so the eigenfunctions allows us to analyze functions living on Ω.
- Next, we will discuss how to represent a signal sampled on a regular lattice (or grids), i.e., a discrete signal instead of a signal in the continuum $[0, \ell]$.

Remarks . . .

- We see that in either BCs, $\{\lambda_n\}_{n=1}^{\infty}$ contains *geometric information* of the domain $\Omega = (0, \ell)$.
- For instance, the size of the first eigenvalue, $\lambda_1 = (\pi/\ell)^2$ tells us the volume of Ω (i.e., the length ℓ of Ω in 1D).
- Under our assumption of constant tension and constant density,

$$\text{small } \lambda_1 \iff \text{long } \ell$$
$$\text{large } \lambda_1 \iff \text{short } \ell$$

- Furthermore, the set $\{\varphi_n\}_{n=1}^{\infty}$ forms an orthonormal basis for $L^2(\Omega)$, so the eigenfunctions allows us to analyze functions living on Ω.
- Next, we will discuss how to represent a signal sampled on a regular lattice (or grids), i.e., a discrete signal instead of a signal in the continuum $[0, \ell]$.

Remarks . . .

- We see that in either BCs, $\{\lambda_n\}_{n=1}^{\infty}$ contains *geometric information* of the domain $\Omega = (0, \ell)$.
- For instance, the size of the first eigenvalue, $\lambda_1 = (\pi/\ell)^2$ tells us the volume of Ω (i.e., the length ℓ of Ω in 1D).
- Under our assumption of constant tension and constant density,

$$\text{small } \lambda_1 \iff \text{long } \ell$$
$$\text{large } \lambda_1 \iff \text{short } \ell$$

- Furthermore, the set $\{\varphi_n\}_{n=1}^{\infty}$ forms an orthonormal basis for $L^2(\Omega)$, so the eigenfunctions allows us to analyze functions living on Ω.
- Next, we will discuss how to represent a signal sampled on a regular lattice (or grids), i.e., a discrete signal instead of a signal in the continuum $[0, \ell]$.

Remarks . . .

- We see that in either BCs, $\{\lambda_n\}_{n=1}^{\infty}$ contains *geometric information* of the domain $\Omega = (0, \ell)$.
- For instance, the size of the first eigenvalue, $\lambda_1 = (\pi/\ell)^2$ tells us the volume of Ω (i.e., the length ℓ of Ω in 1D).
- Under our assumption of constant tension and constant density,

$$\text{small } \lambda_1 \iff \text{long } \ell$$
$$\text{large } \lambda_1 \iff \text{short } \ell$$

- Furthermore, the set $\{\varphi_n\}_{n=1}^{\infty}$ forms an orthonormal basis for $L^2(\Omega)$, so the eigenfunctions allows us to analyze functions living on Ω.
- Next, we will discuss how to represent a signal sampled on a regular lattice (or grids), i.e., a discrete signal instead of a signal in the continuum $[0, \ell]$.

Outline

Outline

Discrete Signal Representation

- Let $\boldsymbol{f} = (f_1, \dots, f_n)^{\mathsf{T}} \in \mathbb{R}^n$ be a vector representing a digital data (e.g., a segment of left channel of your favorite song in an mp3 file).
- We will *not* discuss the important *Analog-to-Digital Conversion* (including quantization) in this lecture due to the time limitation.
- As an example music signal, here is the one with $n = 800,791$ for a little more than 18 seconds. The sampling rate is the standard 44.1 kHz (i.e., 44,100 samples per second).

Discrete Signal Representation

- Let $\boldsymbol{f} = (f_1, \ldots, f_n)^{\mathsf{T}} \in \mathbb{R}^n$ be a vector representing a digital data (e.g., a segment of left channel of your favorite song in an mp3 file).
- We will *not* discuss the important *Analog-to-Digital Conversion* (including quantization) in this lecture due to the time limitation.
- As an example music signal, here is the one with $n = 800,791$ for a little more than 18 seconds. The sampling rate is the standard 44.1 kHz (i.e., 44,100 samples per second).

Discrete Signal Representation

- Let $\boldsymbol{f} = (f_1, \ldots, f_n)^\mathsf{T} \in \mathbb{R}^n$ be a vector representing a digital data (e.g., a segment of left channel of your favorite song in an mp3 file).
- We will *not* discuss the important *Analog-to-Digital Conversion* (including quantization) in this lecture due to the time limitation.
- As an example music signal, here is the one with $n = 800{,}791$ for a little more than 18 seconds. The sampling rate is the standard 44.1 kHz (i.e., 44,100 samples per second).

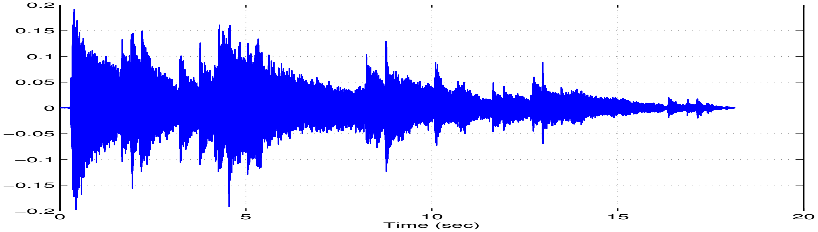


Figure: Steely Dan's Aja: intro part only ≈ 18 sec.

Discrete Signal Representation

- Let $\boldsymbol{f} = (f_1, \ldots, f_n)^\mathsf{T} \in \mathbb{R}^n$ be a vector representing a digital data (e.g., a segment of left channel of your favorite song in an mp3 file).
- We will *not* discuss the important *Analog-to-Digital Conversion* (including quantization) in this lecture due to the time limitation.
- As an example music signal, here is the one with $n = 800,791$ for a little more than 18 seconds. The sampling rate is the standard 44.1 kHz (i.e., 44,100 samples per second).

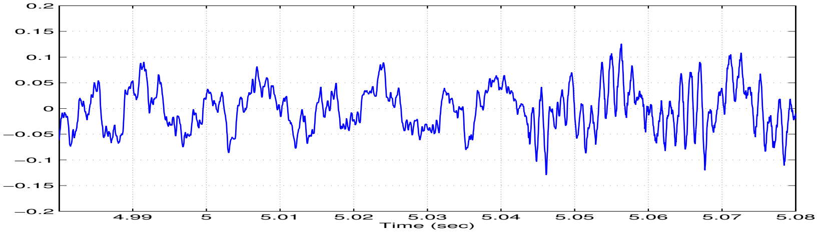


Figure: Zoom up for the period of 0.1 sec.

Discrete Signal Representation

- Let $\boldsymbol{f} = (f_1, \ldots, f_n)^{\mathsf{T}} \in \mathbb{R}^n$ be a vector representing a digital data (e.g., a segment of left channel of your favorite song in an mp3 file).
- We will *not* discuss the important *Analog-to-Digital Conversion* (including quantization) in this lecture due to the time limitation.
- As an example music signal, here is the one with $n = 800,791$ for a little more than 18 seconds. The sampling rate is the standard 44.1 kHz (i.e., 44,100 samples per second).

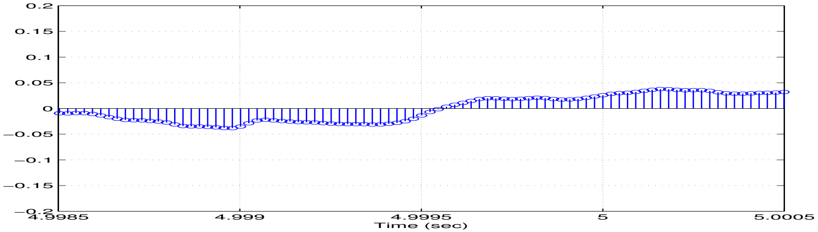


Figure: Further zoom up for the period of 0.002 sec.

Discrete Signal Representation

- Let $\boldsymbol{f} = (f_1, \ldots, f_n)^{\mathsf{T}} \in \mathbb{R}^n$ be a vector representing a digital data (e.g., a segment of left channel of your favorite song in an mp3 file).
- We will *not* discuss the important *Analog-to-Digital Conversion* (including quantization) in this lecture due to the time limitation.
- As an example music signal, here is the one with $n = 800,791$ for a little more than 18 seconds. The sampling rate is the standard 44.1 kHz (i.e., 44,100 samples per second).

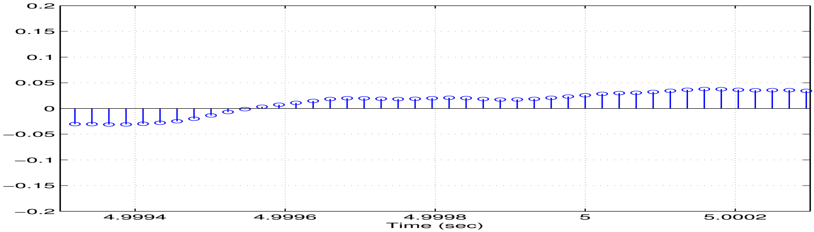


Figure: Further zoom up for the period of 0.001 sec.

Discrete Signal Representation

- Let $\boldsymbol{f} = (f_1, \ldots, f_n)^{\mathsf{T}} \in \mathbb{R}^n$ be a vector representing a digital data (e.g., a segment of left channel of your favorite song in an mp3 file).
- We will *not* discuss the important *Analog-to-Digital Conversion* (including quantization) in this lecture due to the time limitation.
- As an example music signal, here is the one with $n = 800,791$ for a little more than 18 seconds. The sampling rate is the standard 44.1 kHz (i.e., 44,100 samples per second).

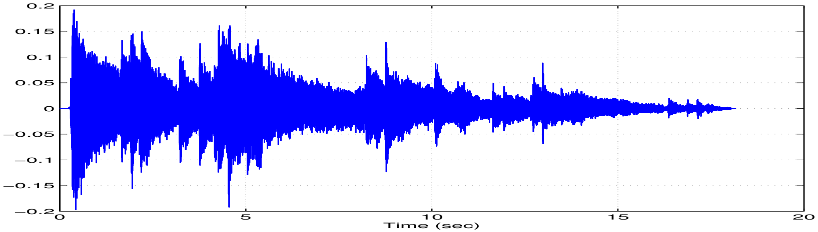


Figure: Aja Intro

Discrete Signal Representation . . .

- Such a vector $\boldsymbol{f} \in \mathbb{R}^n$ can be *represented exactly* using the linear combination of the standard (or canonical) basis of $\mathbb{R}^n$:

$$\boldsymbol{f} = (f_1, \ldots, f_n)^{\mathsf{T}} = f_1 \boldsymbol{e}_1 + \cdots + f_n \boldsymbol{e}_n,$$

where $\boldsymbol{e}_k = (0, \ldots, 0, \underset{\substack{\uparrow \\ k}}{1}, 0, \ldots, 0)^{\mathsf{T}},\ k = 1, \ldots, n.$

- We can write the above in the matrix-vector form:

$$\boldsymbol{f} = I_n \boldsymbol{f},$$

where $I_n := [\boldsymbol{e}_1 | \boldsymbol{e}_2 | \cdots | \boldsymbol{e}_n] \in \mathbb{R}^{n \times n}$, the $n \times n$ *identity matrix*.

- Suppose we want to approximate/compress the original signal with $n' (\ll n)$ samples.
- We will discuss two very simple approximation methods below.

Discrete Signal Representation . . .

- Such a vector $\boldsymbol{f} \in \mathbb{R}^n$ can be *represented exactly* using the linear combination of the standard (or canonical) basis of $\mathbb{R}^n$:

$$\boldsymbol{f} = (f_1, \ldots, f_n)^\mathsf{T} = f_1 \boldsymbol{e}_1 + \cdots + f_n \boldsymbol{e}_n,$$

  where $\boldsymbol{e}_k = (0, \ldots, 0, \underset{\substack{\uparrow \\ k}}{1}, 0, \ldots, 0)^\mathsf{T}, \; k = 1, \ldots, n.$

- We can write the above in the matrix-vector form:

$$\boldsymbol{f} = I_n \boldsymbol{f},$$

  where $I_n := [\boldsymbol{e}_1 | \boldsymbol{e}_2 | \cdots | \boldsymbol{e}_n] \in \mathbb{R}^{n \times n}$, the $n \times n$ *identity matrix*.

- Suppose we want to approximate/compress the original signal with $n' (\ll n)$ samples.
- We will discuss two very simple approximation methods below.

Discrete Signal Representation . . .

- Such a vector $\boldsymbol{f} \in \mathbb{R}^n$ can be *represented exactly* using the linear combination of the standard (or canonical) basis of $\mathbb{R}^n$:

$$\boldsymbol{f} = (f_1, \dots, f_n)^{\mathsf{T}} = f_1 \boldsymbol{e}_1 + \cdots + f_n \boldsymbol{e}_n,$$

  where $\boldsymbol{e}_k = (0, \dots, 0, \underset{\substack{\uparrow \\ k}}{1}, 0, \dots, 0)^{\mathsf{T}},\ k = 1, \dots, n.$

- We can write the above in the matrix-vector form:

$$\boldsymbol{f} = I_n \boldsymbol{f},$$

  where $I_n := [\boldsymbol{e}_1 | \boldsymbol{e}_2 | \cdots | \boldsymbol{e}_n] \in \mathbb{R}^{n \times n}$, the $n \times n$ *identity matrix*.

- Suppose we want to approximate/compress the original signal with $n' (\ll n)$ samples.
- We will discuss two very simple approximation methods below.

Discrete Signal Representation . . .

- Such a vector $\boldsymbol{f} \in \mathbb{R}^n$ can be *represented exactly* using the linear combination of the standard (or canonical) basis of $\mathbb{R}^n$:

$$\boldsymbol{f} = (f_1, \ldots, f_n)^\mathsf{T} = f_1 \boldsymbol{e}_1 + \cdots + f_n \boldsymbol{e}_n,$$

  where $\boldsymbol{e}_k = (0, \ldots, 0, \underset{\substack{\uparrow \\ k}}{1}, 0, \ldots, 0)^\mathsf{T}, \; k = 1, \ldots, n.$

- We can write the above in the matrix-vector form:

$$\boldsymbol{f} = I_n \boldsymbol{f},$$

  where $I_n := [\boldsymbol{e}_1 | \boldsymbol{e}_2 | \cdots | \boldsymbol{e}_n] \in \mathbb{R}^{n \times n}$, the $n \times n$ *identity matrix*.

- Suppose we want to approximate/compress the original signal with $n' (\ll n)$ samples.
- We will discuss two very simple approximation methods below.

I. Linear Approximation

Only retain the first n' coordinates, i.e., set $f_k = 0$ for $k > n'$.

I. Linear Approximation

Only retain the first n' coordinates, i.e., set $f_k = 0$ for $k > n'$.

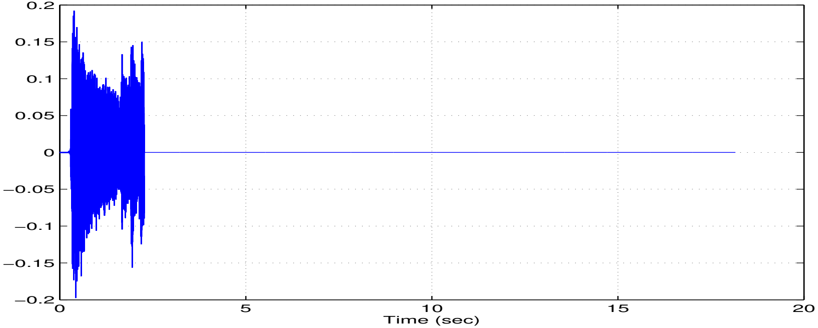


Figure: The first $n' = 100{,}000$ coefficients are retained (only 12.5% of the whole coefficients)

I. Linear Approximation

Only retain the first n' coordinates, i.e., set $f_k = 0$ for $k > n'$.

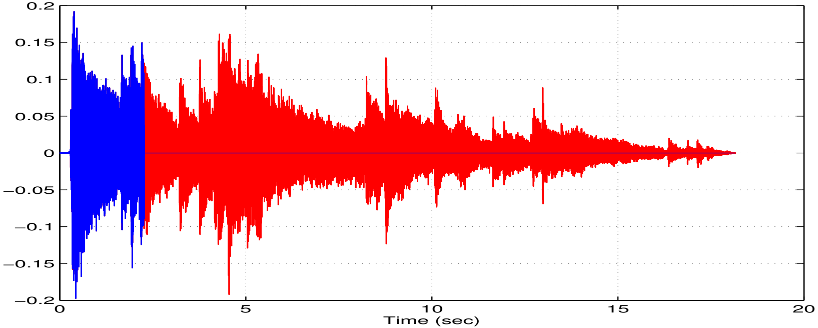


Figure: The red portion represents lost samples

I. Linear Approximation

Only retain the first n' coordinates, i.e., set $f_k = 0$ for $k > n'$.

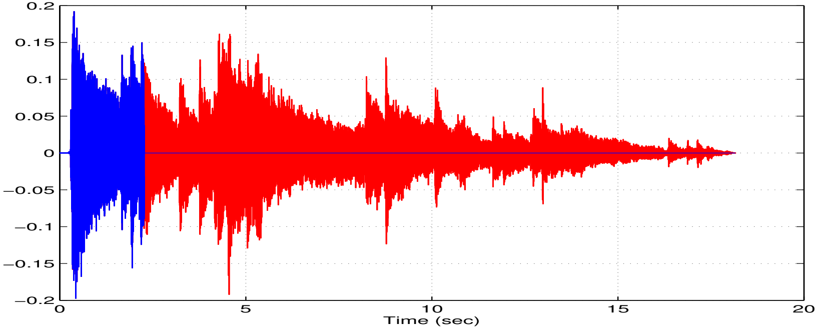


Figure: The red portion represents lost samples

Clearly, this approximation/compression strategy is not efficient for a signal represented in the standard basis!

Aja: the first 100,000 samples

II. Nonlinear Approximation

Sort the absolute value of the coefficients in a nondecreasing order, i.e., $|f_{(1)}| \geq |f_{(2)}| \geq \cdots \geq |f_{(n)}|$; then set $f_{(k)}$ for $k > n'$.

II. Nonlinear Approximation

Sort the absolute value of the coefficients in a nondecreasing order, i.e., $|f_{(1)}| \geq |f_{(2)}| \geq \cdots \geq |f_{(n)}|$; then set $f_{(k)}$ for $k > n'$.

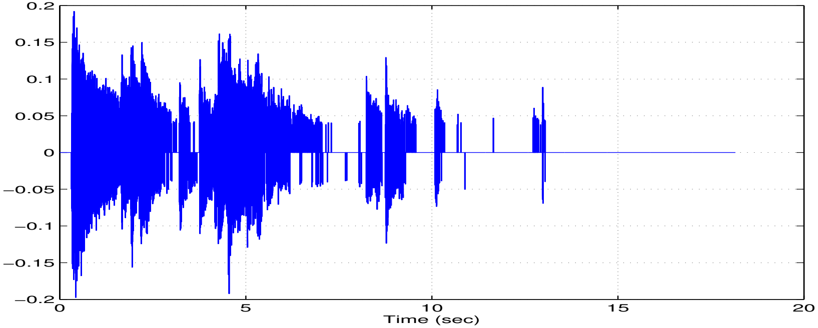


Figure: The 100,000 largest coefficients are retained (only 12.5% of the whole coefficients)

II. Nonlinear Approximation

Sort the absolute value of the coefficients in a nondecreasing order, i.e., $|f_{(1)}| \geq |f_{(2)}| \geq \cdots \geq |f_{(n)}|$; then set $f_{(k)}$ for $k > n'$.

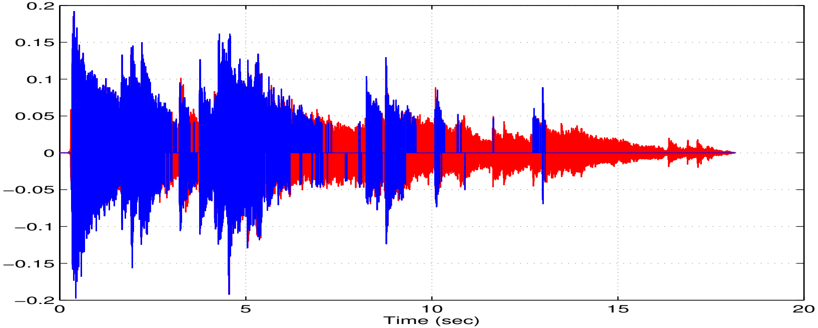


Figure: The red portion represents lost samples

II. Nonlinear Approximation

Sort the absolute value of the coefficients in a nondecreasing order, i.e., $|f_{(1)}| \geq |f_{(2)}| \geq \cdots \geq |f_{(n)}|$; then set $f_{(k)}$ for $k > n'$.

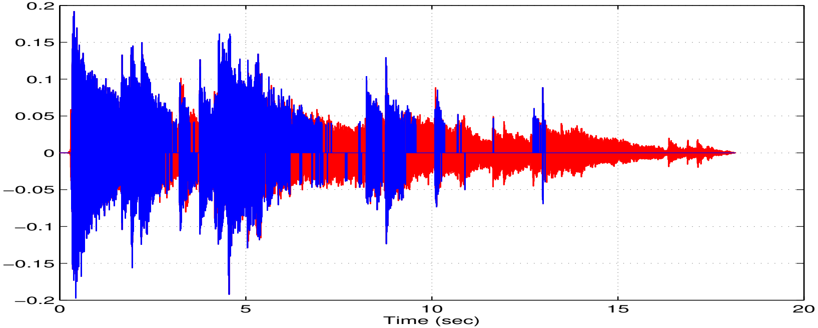


Figure: The red portion represents lost samples

This nonlinear approximation is a bit better than the linear one; yet it is still not good!

Aja: the largest 100,000 samples

Signal Representation via a General Basis

- Suppose there is another *basis* of $\mathbb{R}^n$, say, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$.
- A set of vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$ forms a *basis* of $\mathbb{R}^n$ $\overset{\text{def.}}{\iff}$ $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ are linearly independent and *any* vector $\boldsymbol{f} \in \mathbb{R}^n$ can be written as a linear combination of these basis vectors.
- Let the linear combination coefficients be $\{c_1, \ldots, c_n\}$ such that

$$\boldsymbol{f} = c_1 \boldsymbol{u}_1 + \cdots + c_n \boldsymbol{u}_n = U\boldsymbol{c},$$

  where $U = [\boldsymbol{u}_1 | \cdots | \boldsymbol{u}_n] \in \mathbb{R}^{n \times n}$ and $\boldsymbol{c} = (c_1, \ldots, c_n)^{\mathsf{T}} \in \mathbb{R}^n$ are called the *basis matrix* and the *expansion coefficient vector* of $\boldsymbol{f}$ relative to U, respectively.
- Given an input signal $\boldsymbol{f} \in \mathbb{R}^n$ and a basis $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$, how to compute the expansion coefficients $\{c_1, \ldots, c_n\}$?
- In general, one needs to solve the linear system of equation $\boldsymbol{f} = U\boldsymbol{c}$ numerically, i.e., $\boldsymbol{c} = U^{-1}\boldsymbol{f}$. Normally, solving such a system numerically (or equivalently computing U^{-1} and multiplying it to $\boldsymbol{f}$) costs $O(n^3)$, very expensive, in particular, for large n.

Signal Representation via a General Basis

- Suppose there is another *basis* of $\mathbb{R}^n$, say, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$.
- A set of vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$ forms a *basis* of $\mathbb{R}^n$ $\overset{\text{def.}}{\Longleftrightarrow}$ $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ are linearly independent and *any* vector $\boldsymbol{f} \in \mathbb{R}^n$ can be written as a linear combination of these basis vectors.
- Let the linear combination coefficients be $\{c_1, \ldots, c_n\}$ such that

$$\boldsymbol{f} = c_1 \boldsymbol{u}_1 + \cdots + c_n \boldsymbol{u}_n = U\boldsymbol{c},$$

where $U = [\boldsymbol{u}_1 | \cdots | \boldsymbol{u}_n] \in \mathbb{R}^{n \times n}$ and $\boldsymbol{c} = (c_1, \ldots, c_n)^{\mathsf{T}} \in \mathbb{R}^n$ are called the *basis matrix* and the *expansion coefficient vector* of $\boldsymbol{f}$ relative to U, respectively.
- Given an input signal $\boldsymbol{f} \in \mathbb{R}^n$ and a basis $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$, how to compute the expansion coefficients $\{c_1, \ldots, c_n\}$?
- In general, one needs to solve the linear system of equation $\boldsymbol{f} = U\boldsymbol{c}$ numerically, i.e., $\boldsymbol{c} = U^{-1}\boldsymbol{f}$. Normally, solving such a system numerically (or equivalently computing U^{-1} and multiplying it to $\boldsymbol{f}$) costs $O(n^3)$, very expensive, in particular, for large n.

Signal Representation via a General Basis

- Suppose there is another *basis* of $\mathbb{R}^n$, say, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$.
- A set of vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$ forms a *basis* of $\mathbb{R}^n$ $\overset{\text{def.}}{\Longleftrightarrow}$ $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ are linearly independent and *any* vector $\boldsymbol{f} \in \mathbb{R}^n$ can be written as a linear combination of these basis vectors.
- Let the linear combination coefficients be $\{c_1, \ldots, c_n\}$ such that

$$\boldsymbol{f} = c_1 \boldsymbol{u}_1 + \cdots + c_n \boldsymbol{u}_n = U\boldsymbol{c},$$

  where $U = [\boldsymbol{u}_1 | \cdots | \boldsymbol{u}_n] \in \mathbb{R}^{n \times n}$ and $\boldsymbol{c} = (c_1, \ldots, c_n)^{\mathsf{T}} \in \mathbb{R}^n$ are called the *basis matrix* and the *expansion coefficient vector* of $\boldsymbol{f}$ relative to U, respectively.
- Given an input signal $\boldsymbol{f} \in \mathbb{R}^n$ and a basis $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$, how to compute the expansion coefficients $\{c_1, \ldots, c_n\}$?
- In general, one needs to solve the linear system of equation $\boldsymbol{f} = U\boldsymbol{c}$ numerically, i.e., $\boldsymbol{c} = U^{-1}\boldsymbol{f}$. Normally, solving such a system numerically (or equivalently computing U^{-1} and multiplying it to $\boldsymbol{f}$) costs $O(n^3)$, very expensive, in particular, for large n.

Signal Representation via a General Basis

- Suppose there is another *basis* of $\mathbb{R}^n$, say, $\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n\} \subset \mathbb{R}^n$.
- A set of vectors $\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n\} \subset \mathbb{R}^n$ forms a *basis* of $\mathbb{R}^n$ $\stackrel{\text{def.}}{\Longleftrightarrow}$ $\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n\}$ are linearly independent and *any* vector $\boldsymbol{f} \in \mathbb{R}^n$ can be written as a linear combination of these basis vectors.
- Let the linear combination coefficients be $\{c_1,\ldots,c_n\}$ such that

$$\boldsymbol{f} = c_1\boldsymbol{u}_1 + \cdots + c_n\boldsymbol{u}_n = U\boldsymbol{c},$$

  where $U = [\boldsymbol{u}_1|\cdots|\boldsymbol{u}_n] \in \mathbb{R}^{n\times n}$ and $\boldsymbol{c} = (c_1,\ldots,c_n)^{\mathsf{T}} \in \mathbb{R}^n$ are called the *basis matrix* and the *expansion coefficient vector* of $\boldsymbol{f}$ relative to U, respectively.
- Given an input signal $\boldsymbol{f} \in \mathbb{R}^n$ and a basis $\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n\} \subset \mathbb{R}^n$, how to compute the expansion coefficients $\{c_1,\ldots,c_n\}$?
- In general, one needs to solve the linear system of equation $\boldsymbol{f} = U\boldsymbol{c}$ numerically, i.e., $\boldsymbol{c} = U^{-1}\boldsymbol{f}$. Normally, solving such a system numerically (or equivalently computing U^{-1} and multiplying it to $\boldsymbol{f}$) costs $O(n^3)$, very expensive, in particular, for large n.

Signal Representation via a General Basis

- Suppose there is another *basis* of $\mathbb{R}^n$, say, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$.
- A set of vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$ forms a *basis* of $\mathbb{R}^n$ $\overset{\text{def.}}{\Longleftrightarrow}$ $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ are linearly independent and *any* vector $\boldsymbol{f} \in \mathbb{R}^n$ can be written as a linear combination of these basis vectors.
- Let the linear combination coefficients be $\{c_1, \ldots, c_n\}$ such that

$$\boldsymbol{f} = c_1 \boldsymbol{u}_1 + \cdots + c_n \boldsymbol{u}_n = U\boldsymbol{c},$$

  where $U = [\boldsymbol{u}_1 | \cdots | \boldsymbol{u}_n] \in \mathbb{R}^{n \times n}$ and $\boldsymbol{c} = (c_1, \ldots, c_n)^{\mathsf{T}} \in \mathbb{R}^n$ are called the *basis matrix* and the *expansion coefficient vector* of $\boldsymbol{f}$ relative to U, respectively.
- Given an input signal $\boldsymbol{f} \in \mathbb{R}^n$ and a basis $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \subset \mathbb{R}^n$, how to compute the expansion coefficients $\{c_1, \ldots, c_n\}$?
- In general, one needs to solve the linear system of equation $\boldsymbol{f} = U\boldsymbol{c}$ numerically, i.e., $\boldsymbol{c} = U^{-1}\boldsymbol{f}$. Normally, solving such a system numerically (or equivalently computing U^{-1} and multiplying it to $\boldsymbol{f}$) costs $O(n^3)$, very expensive, in particular, for large n.

Signal Representation via an Orthonormal Basis

- However, if the basis vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then we can reduce the computational cost to $O(n^2)$.
- Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{y}^\mathsf{T}\boldsymbol{x} = x_1 y_1 + \cdots + x_n y_n$ be the standard *inner (or dot or scalar) product* in $\mathbb{R}^n$.
- The orthonormality of the vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ means

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = \delta_{ij} \text{ (Kronecker's delta)} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ is an orthonormal basis of $\mathbb{R}^n$, then we simplify the computation of the expansion coefficients:

$$\begin{aligned} \langle \boldsymbol{f}, \boldsymbol{u}_j \rangle &= \langle c_1 \boldsymbol{u}_1 + \cdots + c_j \boldsymbol{u}_j + \cdots + c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= \langle c_1 \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + \langle c_j \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + \langle c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \langle \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + c_j \langle \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + c_n \langle \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \cdot 0 + \cdots + c_j \cdot 1 + \cdots + c_n \cdot 0 \\ &= c_j \end{aligned}$$

Signal Representation via an Orthonormal Basis

- However, if the basis vectors $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then we can reduce the computational cost to $O(n^2)$.
- Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{y}^\mathsf{T} \boldsymbol{x} = x_1 y_1 + \cdots + x_n y_n$ be the standard *inner (or dot or scalar) product* in $\mathbb{R}^n$.
- The orthonormality of the vectors $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_n\}$ means

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = \delta_{ij} \text{ (Kronecker's delta)} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

- If $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_n\}$ is an orthonormal basis of $\mathbb{R}^n$, then we simplify the computation of the expansion coefficients:

$$\begin{aligned} \langle \boldsymbol{f}, \boldsymbol{u}_j \rangle &= \langle c_1 \boldsymbol{u}_1 + \cdots + c_j \boldsymbol{u}_j + \cdots + c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= \langle c_1 \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + \langle c_j \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + \langle c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \langle \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + c_j \langle \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + c_n \langle \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \cdot 0 + \cdots + c_j \cdot 1 + \cdots + c_n \cdot 0 \\ &= c_j \end{aligned}$$

Signal Representation via an Orthonormal Basis

- However, if the basis vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then we can reduce the computational cost to $O(n^2)$.
- Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{y}^{\mathsf{T}} \boldsymbol{x} = x_1 y_1 + \cdots + x_n y_n$ be the standard *inner (or dot or scalar) product* in $\mathbb{R}^n$.
- The orthonormality of the vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ means

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = \delta_{ij} \text{ (Kronecker's delta)} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ is an orthonormal basis of $\mathbb{R}^n$, then we simplify the computation of the expansion coefficients:

$$\begin{aligned} \langle \boldsymbol{f}, \boldsymbol{u}_j \rangle &= \langle c_1 \boldsymbol{u}_1 + \cdots + c_j \boldsymbol{u}_j + \cdots + c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= \langle c_1 \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + \langle c_j \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + \langle c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \langle \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + c_j \langle \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + c_n \langle \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \cdot 0 + \cdots + c_j \cdot 1 + \cdots + c_n \cdot 0 \\ &= c_j \end{aligned}$$

Signal Representation via an Orthonormal Basis

- However, if the basis vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then we can reduce the computational cost to $O(n^2)$.
- Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{y}^{\mathsf{T}} \boldsymbol{x} = x_1 y_1 + \cdots + x_n y_n$ be the standard *inner (or dot or scalar) product* in $\mathbb{R}^n$.
- The orthonormality of the vectors $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ means

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = \delta_{ij} \text{ (Kronecker's delta)} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ is an orthonormal basis of $\mathbb{R}^n$, then we simplify the computation of the expansion coefficients:

$$\begin{aligned} \langle \boldsymbol{f}, \boldsymbol{u}_j \rangle &= \langle c_1 \boldsymbol{u}_1 + \cdots + c_j \boldsymbol{u}_j + \cdots + c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= \langle c_1 \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + \langle c_j \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + \langle c_n \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \langle \boldsymbol{u}_1, \boldsymbol{u}_j \rangle + \cdots + c_j \langle \boldsymbol{u}_j, \boldsymbol{u}_j \rangle + \cdots + c_n \langle \boldsymbol{u}_n, \boldsymbol{u}_j \rangle \\ &= c_1 \cdot 0 + \cdots + c_j \cdot 1 + \cdots + c_n \cdot 0 \\ &= c_j \end{aligned}$$

Signal Representation via an Orthonormal Basis . . .

- If $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then the basis matrix U is called an orthogonal matrix in $\mathbb{R}^n$, i.e., $U^{\mathsf{T}}U = UU^{\mathsf{T}} = I_n$.
- Therefore, there is no need to compute U^{-1} since $U^{-1} = U^{\mathsf{T}}$ if U is an orthogonal matrix.
- Consequently, we can compute the expansion coefficient vector $\boldsymbol{c}$ by $\boldsymbol{c} = U^{\mathsf{T}}\boldsymbol{f}$!!
- This is a simple matrix-vector multiplication (with a matrix transpose operation), which costs $O(n^2)$ in general.
- Clearly, the standard basis of $\mathbb{R}^n$, $\{\boldsymbol{e}_1, \dots, \boldsymbol{e}_n\}$, is an orthonormal basis but these basis vectors are very *localized*.
- Other examples of orthonormal bases include: Discrete Fourier Basis; *Discrete Cosine Basis*; Discrete Sine Basis; Discrete Wavelet Basis; *Principal Component Analysis (PCA) Basis* (a.k.a. *Karhunen-Loève Basis*), . . .

Signal Representation via an Orthonormal Basis . . .

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then the basis matrix U is called an orthogonal matrix in $\mathbb{R}^n$, i.e., $U^{\mathsf{T}}U = UU^{\mathsf{T}} = I_n$.
- Therefore, there is no need to compute U^{-1} since $U^{-1} = U^{\mathsf{T}}$ if U is an orthogonal matrix.
- Consequently, we can compute the expansion coefficient vector $\boldsymbol{c}$ by $\boldsymbol{c} = U^{\mathsf{T}}\boldsymbol{f}$!!
- This is a simple matrix-vector multiplication (with a matrix transpose operation), which costs $O(n^2)$ in general.
- Clearly, the standard basis of $\mathbb{R}^n$, $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, is an orthonormal basis but these basis vectors are very *localized*.
- Other examples of orthonormal bases include: Discrete Fourier Basis; *Discrete Cosine Basis*; Discrete Sine Basis; Discrete Wavelet Basis; *Principal Component Analysis (PCA) Basis* (a.k.a. *Karhunen-Loève Basis*), . . .

Signal Representation via an Orthonormal Basis . . .

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then the basis matrix U is called an orthogonal matrix in $\mathbb{R}^n$, i.e., $U^{\mathsf{T}} U = U U^{\mathsf{T}} = I_n$.
- Therefore, there is no need to compute U^{-1} since $U^{-1} = U^{\mathsf{T}}$ if U is an orthogonal matrix.
- Consequently, we can compute the expansion coefficient vector $\boldsymbol{c}$ by $\boldsymbol{c} = U^{\mathsf{T}} \boldsymbol{f}$!!
- This is a simple matrix-vector multiplication (with a matrix transpose operation), which costs $O(n^2)$ in general.
- Clearly, the standard basis of $\mathbb{R}^n$, $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, is an orthonormal basis but these basis vectors are very *localized*.
- Other examples of orthonormal bases include: Discrete Fourier Basis; *Discrete Cosine Basis*; Discrete Sine Basis; Discrete Wavelet Basis; *Principal Component Analysis (PCA) Basis* (a.k.a. *Karhunen-Loève Basis*), . . .

Signal Representation via an Orthonormal Basis . . .

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then the basis matrix U is called an orthogonal matrix in $\mathbb{R}^n$, i.e., $U^{\mathsf{T}} U = U U^{\mathsf{T}} = I_n$.
- Therefore, there is no need to compute U^{-1} since $U^{-1} = U^{\mathsf{T}}$ if U is an orthogonal matrix.
- Consequently, we can compute the expansion coefficient vector $\boldsymbol{c}$ by $\boldsymbol{c} = U^{\mathsf{T}} \boldsymbol{f}$!!
- This is a simple matrix-vector multiplication (with a matrix transpose operation), which costs $O(n^2)$ in general.
- Clearly, the standard basis of $\mathbb{R}^n$, $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, is an orthonormal basis but these basis vectors are very *localized*.
- Other examples of orthonormal bases include: Discrete Fourier Basis; *Discrete Cosine Basis*; Discrete Sine Basis; Discrete Wavelet Basis; *Principal Component Analysis (PCA) Basis* (a.k.a. *Karhunen-Loève Basis*), . . .

Signal Representation via an Orthonormal Basis . . .

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then the basis matrix U is called an orthogonal matrix in $\mathbb{R}^n$, i.e., $U^{\mathsf{T}}U = UU^{\mathsf{T}} = I_n$.
- Therefore, there is no need to compute U^{-1} since $U^{-1} = U^{\mathsf{T}}$ if U is an orthogonal matrix.
- Consequently, we can compute the expansion coefficient vector $\boldsymbol{c}$ by $\boldsymbol{c} = U^{\mathsf{T}}\boldsymbol{f}$!!
- This is a simple matrix-vector multiplication (with a matrix transpose operation), which costs $O(n^2)$ in general.
- Clearly, the standard basis of $\mathbb{R}^n$, $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, is an orthonormal basis but these basis vectors are very *localized*.
- Other examples of orthonormal bases include: Discrete Fourier Basis; *Discrete Cosine Basis*; Discrete Sine Basis; Discrete Wavelet Basis; *Principal Component Analysis (PCA) Basis* (a.k.a. *Karhunen-Loève Basis*), . . .

Signal Representation via an Orthonormal Basis . . .

- If $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ form an orthonormal basis of $\mathbb{R}^n$, then the basis matrix U is called an orthogonal matrix in $\mathbb{R}^n$, i.e., $U^\mathsf{T} U = U U^\mathsf{T} = I_n$.
- Therefore, there is no need to compute U^{-1} since $U^{-1} = U^\mathsf{T}$ if U is an orthogonal matrix.
- Consequently, we can compute the expansion coefficient vector $\boldsymbol{c}$ by $\boldsymbol{c} = U^\mathsf{T} \boldsymbol{f}$!!
- This is a simple matrix-vector multiplication (with a matrix transpose operation), which costs $O(n^2)$ in general.
- Clearly, the standard basis of $\mathbb{R}^n$, $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, is an orthonormal basis but these basis vectors are very *localized*.
- Other examples of orthonormal bases include: Discrete Fourier Basis; *Discrete Cosine Basis*; Discrete Sine Basis; Discrete Wavelet Basis; *Principal Component Analysis (PCA) Basis* (a.k.a. *Karhunen-Loève Basis*), . . .

Outline

A Few Words on Bases in the Continuum

- Let's briefly return to the continuous setting of the 1D wave equation with the Neumann BC.
- Then, recall the basis functions in this case are:

$$\varphi_k(x) := \begin{cases} \frac{1}{\sqrt{\ell}} & \text{if } k = 0; \\ \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x}{\ell}\right), & \text{if } k > 0. \end{cases}$$

  where $k \in \mathbb{N}_0 := \{0\} \cup \mathbb{N} = \{0, 1, \ldots\}$ represents the *frequency*.
- Just like $\left\{\sqrt{\frac{2}{\ell}} \sin\left(\frac{k\pi x}{\ell}\right)\right\}_{k\in\mathbb{N}}$, the above $\{\varphi_k(x)\}_{k\in\mathbb{N}_0}$ also form an orthonormal basis of $L^2[0,\ell]$. In other words, every finite energy function defined on $[0,\ell]$ can be written as a linear combination of $\{\varphi_k(x)\}_{k\in\mathbb{N}_0}$.

A Few Words on Bases in the Continuum

- Let's briefly return to the continuous setting of the 1D wave equation with the Neumann BC.
- Then, recall the basis functions in this case are:

$$\varphi_k(x) := \begin{cases} \frac{1}{\sqrt{\ell}} & \text{if } k = 0; \\ \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x}{\ell}\right), & \text{if } k > 0. \end{cases}$$

  where $k \in \mathbb{N}_0 := \{0\} \cup \mathbb{N} = \{0, 1, \ldots\}$ represents the *frequency*.
- Just like $\left\{\sqrt{\frac{2}{\ell}} \sin\left(\frac{k\pi x}{\ell}\right)\right\}_{k\in\mathbb{N}}$, the above $\{\varphi_k(x)\}_{k\in\mathbb{N}_0}$ also form an orthonormal basis of $L^2[0,\ell]$. In other words, every finite energy function defined on $[0,\ell]$ can be written as a linear combination of $\{\varphi_k(x)\}_{k\in\mathbb{N}_0}$.

A Few Words on Bases in the Continuum

- Let's briefly return to the continuous setting of the 1D wave equation with the Neumann BC.
- Then, recall the basis functions in this case are:

$$\varphi_k(x) := \begin{cases} \frac{1}{\sqrt{\ell}} & \text{if } k = 0; \\ \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x}{\ell}\right), & \text{if } k > 0. \end{cases}$$

where $k \in \mathbb{N}_0 := \{0\} \cup \mathbb{N} = \{0, 1, \ldots\}$ represents the *frequency*.

- Just like $\left\{\sqrt{\frac{2}{\ell}} \sin\left(\frac{k\pi x}{\ell}\right)\right\}_{k \in \mathbb{N}}$, the above $\{\varphi_k(x)\}_{k \in \mathbb{N}_0}$ also form an orthonormal basis of $L^2[0, \ell]$. In other words, every finite energy function defined on $[0, \ell]$ can be written as a linear combination of $\{\varphi_k(x)\}_{k \in \mathbb{N}_0}$.

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j + \frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\varphi_{k'}(x)\,\mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\varphi_{k'}(x_j)\Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j)\sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j+\frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\varphi_{k'}(x)\,\mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\varphi_{k'}(x_j)\Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j)\sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact = !!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j+\frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}}\cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}}\cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}}\cdot\sqrt{\frac{2}{n}}\cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\varphi_{k'}(x)\,\mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\varphi_{k'}(x_j)\Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j)\sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j + \frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j + \frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j + \frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\varphi_{k'}(x)\,\mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\varphi_{k'}(x_j)\Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j)\sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j+\frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\, \varphi_{k'}(x)\, \mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\, \varphi_{k'}(x_j) \Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x} \varphi_k(x_j) \sqrt{\Delta x} \varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j+\frac{1}{2}\right)\Delta x$, $j = 0, 1, \dots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\,\varphi_{k'}(x)\,\mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\,\varphi_{k'}(x_j)\Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j)\,\sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j+\frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\, \varphi_{k'}(x)\, \mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\, \varphi_{k'}(x_j) \Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j) \sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j+\frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j+\frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\, \varphi_{k'}(x)\, \mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\, \varphi_{k'}(x_j) \Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j)\, \sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j + \frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j + \frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j + \frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\, \varphi_{k'}(x)\, \mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\, \varphi_{k'}(x_j) \Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x} \varphi_k(x_j)\, \sqrt{\Delta x} \varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

Discrete Cosine Transform (DCT)

- Let's discretize $\varphi_k(x)$ at the *regularly sampled* points $x_j := \left(j + \frac{1}{2}\right)\Delta x$, $j = 0, 1, \ldots, n-1$, where $\Delta x := \frac{\ell}{n}$ is called the *sampling rate*.
- Then, the resulting vector (for $k > 0$) is:

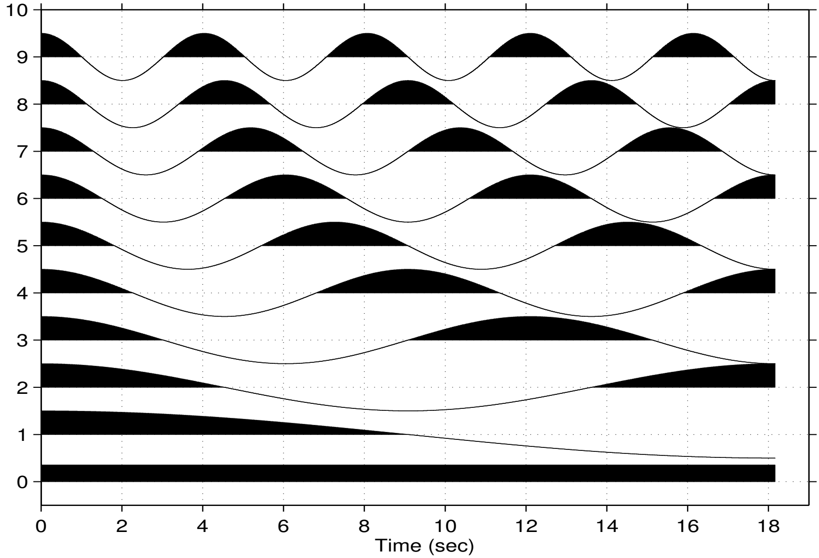
$$\varphi_k(x_j) := \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k x_j}{\ell}\right) = \sqrt{\frac{2}{\ell}} \cos\left(\frac{\pi k (j + \frac{1}{2})}{n}\right) = \frac{1}{\sqrt{\Delta x}} \cdot \sqrt{\frac{2}{n}} \cos\left(\frac{\pi k (j + \frac{1}{2})}{n}\right)$$

- By defining $\boldsymbol{\varphi}_k := \sqrt{\Delta x}\left(\varphi_k(x_0), \cdots, \varphi_k(x_{n-1})\right)^{\mathsf{T}} \in \mathbb{R}^n$, we have:

$$\begin{aligned}
\langle \varphi_k, \varphi_{k'} \rangle_{L^2[0,\ell]} &= \int_0^\ell \varphi_k(x)\,\varphi_{k'}(x)\,\mathrm{d}x \\
&\approx \sum_{j=0}^{n-1} \varphi_k(x_j)\,\varphi_{k'}(x_j)\Delta x \quad \text{The Midpoint Rule!} \\
&= \sum_{j=0}^{n-1} \sqrt{\Delta x}\varphi_k(x_j)\,\sqrt{\Delta x}\varphi_{k'}(x_j) \\
&= \langle \boldsymbol{\varphi}_k, \boldsymbol{\varphi}_{k'} \rangle_{\mathbb{R}^n} = \delta_{kk'}
\end{aligned}$$

Hence, $\approx$ above turns out to be exact $=$!!

The First 10 vectors $\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_9$



Discrete Cosine Transform (DCT) ...

- One can show that $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ form an orthonormal basis of $\mathbb{R}^n$.
- Hence, the Discrete Cosine Basis matrix, $\Phi := [\boldsymbol{\varphi}_0 | \cdots | \boldsymbol{\varphi}_{n-1}] \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.
- Hence, we can write a given vector $\boldsymbol{f}$ as a linear combination of $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., $\boldsymbol{f} = \Phi \boldsymbol{c}$, and the expansion coefficient vector $\boldsymbol{c}$ can be computed by $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$.
- Normally, it would take $O(n^2)$ to compute $\boldsymbol{c}$, but in this case, thanks to the special properties of these basis vectors, one can compute $\boldsymbol{c}$ in $O(n \log n)$ based on the *Fast Fourier Transform* (FFT) algorithm!
- The process of transforming an input vector to the expansion coefficients relative to the Discrete Cosine Basis is referred to as Discrete Cosine Transform (DCT); or DCT-Type II to be more precise.
- See, e.g., G. Strang: "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999, for the details.
- Let's see how the quality of approximation changes if we use $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ instead of $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

Discrete Cosine Transform (DCT) ...

- One can show that $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ form an orthonormal basis of $\mathbb{R}^n$.
- Hence, the Discrete Cosine Basis matrix, $\Phi := [\boldsymbol{\varphi}_0 | \cdots | \boldsymbol{\varphi}_{n-1}] \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.
- Hence, we can write a given vector $\boldsymbol{f}$ as a linear combination of $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., $\boldsymbol{f} = \Phi \boldsymbol{c}$, and the expansion coefficient vector $\boldsymbol{c}$ can be computed by $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$.
- Normally, it would take $O(n^2)$ to compute $\boldsymbol{c}$, but in this case, thanks to the special properties of these basis vectors, one can compute $\boldsymbol{c}$ in $O(n \log n)$ based on the *Fast Fourier Transform* (FFT) algorithm!
- The process of transforming an input vector to the expansion coefficients relative to the Discrete Cosine Basis is referred to as Discrete Cosine Transform (DCT); or DCT-Type II to be more precise.
- See, e.g., G. Strang: "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999, for the details.
- Let's see how the quality of approximation changes if we use $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ instead of $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

Discrete Cosine Transform (DCT) ...

- One can show that $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ form an orthonormal basis of $\mathbb{R}^n$.
- Hence, the Discrete Cosine Basis matrix, $\Phi := [\boldsymbol{\varphi}_0 | \cdots | \boldsymbol{\varphi}_{n-1}] \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.
- Hence, we can write a given vector $\boldsymbol{f}$ as a linear combination of $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., $\boldsymbol{f} = \Phi \boldsymbol{c}$, and the expansion coefficient vector $\boldsymbol{c}$ can be computed by $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$.
- Normally, it would take $O(n^2)$ to compute $\boldsymbol{c}$, but in this case, thanks to the special properties of these basis vectors, one can compute $\boldsymbol{c}$ in $O(n \log n)$ based on the *Fast Fourier Transform* (FFT) algorithm!
- The process of transforming an input vector to the expansion coefficients relative to the Discrete Cosine Basis is referred to as Discrete Cosine Transform (DCT); or DCT-Type II to be more precise.
- See, e.g., G. Strang: "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999, for the details.
- Let's see how the quality of approximation changes if we use $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ instead of $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

Discrete Cosine Transform (DCT) . . .

- One can show that $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ form an orthonormal basis of $\mathbb{R}^n$.
- Hence, the Discrete Cosine Basis matrix, $\Phi := \left[\boldsymbol{\varphi}_0 | \cdots | \boldsymbol{\varphi}_{n-1}\right] \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.
- Hence, we can write a given vector $\boldsymbol{f}$ as a linear combination of $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., $\boldsymbol{f} = \Phi \boldsymbol{c}$, and the expansion coefficient vector $\boldsymbol{c}$ can be computed by $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$.
- Normally, it would take $O(n^2)$ to compute $\boldsymbol{c}$, but in this case, thanks to the special properties of these basis vectors, one can compute $\boldsymbol{c}$ in $O(n \log n)$ based on the *Fast Fourier Transform* (FFT) algorithm!
- The process of transforming an input vector to the expansion coefficients relative to the Discrete Cosine Basis is referred to as Discrete Cosine Transform (DCT); or DCT-Type II to be more precise.
- See, e.g., G. Strang: "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999, for the details.
- Let's see how the quality of approximation changes if we use $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ instead of $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

Discrete Cosine Transform (DCT) . . .

- One can show that $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ form an orthonormal basis of $\mathbb{R}^n$.
- Hence, the Discrete Cosine Basis matrix, $\Phi := \left[\boldsymbol{\varphi}_0 | \cdots | \boldsymbol{\varphi}_{n-1}\right] \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.
- Hence, we can write a given vector $\boldsymbol{f}$ as a linear combination of $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., $\boldsymbol{f} = \Phi \boldsymbol{c}$, and the expansion coefficient vector $\boldsymbol{c}$ can be computed by $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$.
- Normally, it would take $O(n^2)$ to compute $\boldsymbol{c}$, but in this case, thanks to the special properties of these basis vectors, one can compute $\boldsymbol{c}$ in $O(n \log n)$ based on the *Fast Fourier Transform* (FFT) algorithm!
- The process of transforming an input vector to the expansion coefficients relative to the Discrete Cosine Basis is referred to as Discrete Cosine Transform (DCT); or DCT-Type II to be more precise.
- See, e.g., G. Strang: "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999, for the details.
- Let's see how the quality of approximation changes if we use $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ instead of $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

Discrete Cosine Transform (DCT) ...

- One can show that $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ form an orthonormal basis of $\mathbb{R}^n$.
- Hence, the Discrete Cosine Basis matrix, $\Phi := \left[\boldsymbol{\varphi}_0 | \cdots | \boldsymbol{\varphi}_{n-1}\right] \in \mathbb{R}^{n\times n}$ is an orthogonal matrix.
- Hence, we can write a given vector $\boldsymbol{f}$ as a linear combination of $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., $\boldsymbol{f} = \Phi \boldsymbol{c}$, and the expansion coefficient vector $\boldsymbol{c}$ can be computed by $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$.
- Normally, it would take $O(n^2)$ to compute $\boldsymbol{c}$, but in this case, thanks to the special properties of these basis vectors, one can compute $\boldsymbol{c}$ in $O(n \log n)$ based on the *Fast Fourier Transform* (FFT) algorithm!
- The process of transforming an input vector to the expansion coefficients relative to the Discrete Cosine Basis is referred to as Discrete Cosine Transform (DCT); or DCT-Type II to be more precise.
- See, e.g., G. Strang: "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999, for the details.
- Let's see how the quality of approximation changes if we use $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ instead of $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

Discrete Cosine Transform (DCT) . . .

- One can show that $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ form an orthonormal basis of $\mathbb{R}^n$.
- Hence, the Discrete Cosine Basis matrix, $\Phi := [\boldsymbol{\varphi}_0 | \cdots | \boldsymbol{\varphi}_{n-1}] \in \mathbb{R}^{n \times n}$ is an orthogonal matrix.
- Hence, we can write a given vector $\boldsymbol{f}$ as a linear combination of $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$, i.e., $\boldsymbol{f} = \Phi \boldsymbol{c}$, and the expansion coefficient vector $\boldsymbol{c}$ can be computed by $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$.
- Normally, it would take $O(n^2)$ to compute $\boldsymbol{c}$, but in this case, thanks to the special properties of these basis vectors, one can compute $\boldsymbol{c}$ in $O(n \log n)$ based on the *Fast Fourier Transform* (FFT) algorithm!
- The process of transforming an input vector to the expansion coefficients relative to the Discrete Cosine Basis is referred to as Discrete Cosine Transform (DCT); or DCT-Type II to be more precise.
- See, e.g., G. Strang: "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999, for the details.
- Let's see how the quality of approximation changes if we use $\{\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_{n-1}\}$ instead of $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

I. Linear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; and then multiply Φ to the modified coefficient vector.

I. Linear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; and then multiply Φ to the modified coefficient vector.

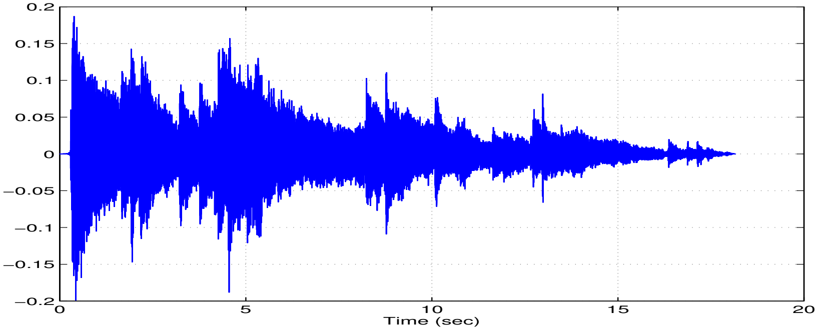


Figure: The first 100,000 DCT coefficients are retained (only 12.5% of the whole coefficients): Aja: the first 100,000 DCT coefficients

I. Linear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; and then multiply Φ to the modified coefficient vector.

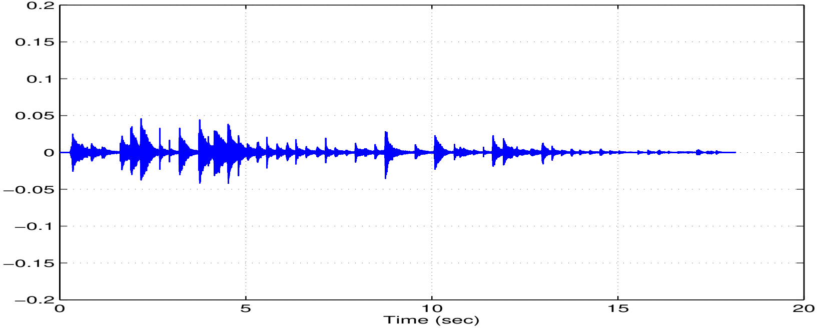


Figure: Diff. from the original: Aja: the last 700,791 DCT coefficients

I. Linear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; and then multiply Φ to the modified coefficient vector.

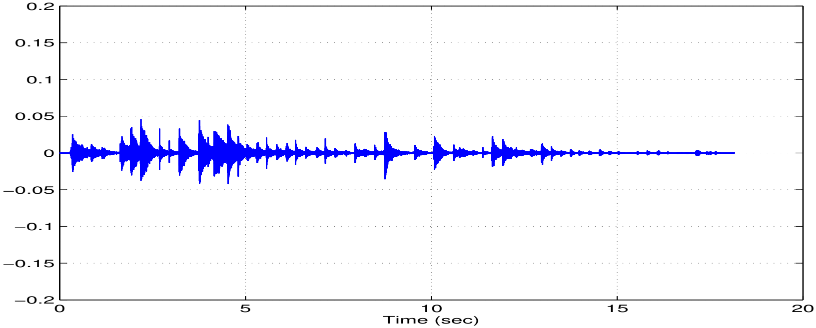


Figure: Diff. from the original: Aja: the last 700,791 DCT coefficients

Clearly, this approximation with DCT is much more efficient than using the standard basis!

II. Nonlinear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; sort the absolute value of the coefficients in a nondecreasing order, i.e., $|c_{(1)}| \geq |c_{(2)}| \geq \cdots \geq |c_{(n)}|$; then set $c_{(k)}$ for $k > n'$; then multiply Φ to the modified coefficient vector.

II. Nonlinear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; sort the absolute value of the coefficients in a nondecreasing order, i.e., $|c_{(1)}| \geq |c_{(2)}| \geq \cdots \geq |c_{(n)}|$; then set $c_{(k)}$ for $k > n'$; then multiply Φ to the modified coefficient vector.

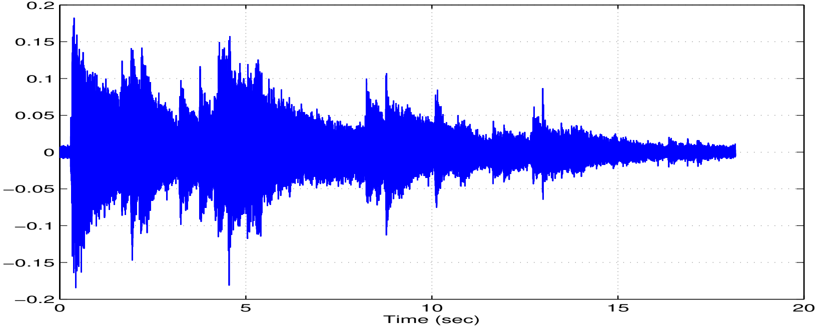


Figure: The 100,000 largest DCT coefficients are retained (only 12.5% of the whole coefficients): Aja: the largest 100,000 DCT coefficients

II. Nonlinear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; sort the absolute value of the coefficients in a nondecreasing order, i.e., $|c_{(1)}| \geq |c_{(2)}| \geq \cdots \geq |c_{(n)}|$; then set $c_{(k)}$ for $k > n'$; then multiply Φ to the modified coefficient vector.

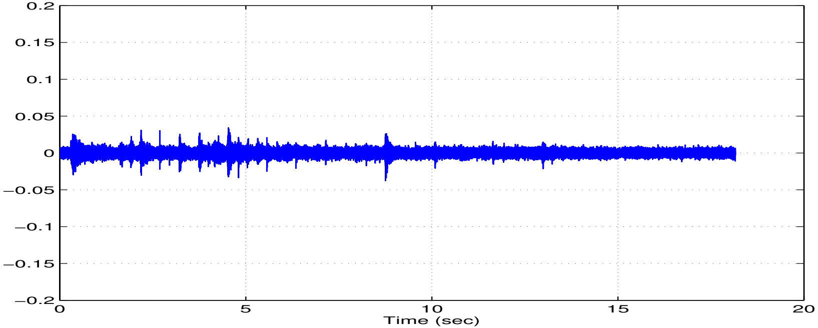


Figure: Diff. from the original: Aja: the smallest 700,791 DCT coefficients

II. Nonlinear Approximation with DCT

Compute $\boldsymbol{c} = \Phi^{\mathsf{T}} \boldsymbol{f}$; set $c_k = 0$ for $k > n'$; sort the absolute value of the coefficients in a nondecreasing order, i.e., $|c_{(1)}| \geq |c_{(2)}| \geq \cdots \geq |c_{(n)}|$; then set $c_{(k)}$ for $k > n'$; then multiply Φ to the modified coefficient vector.

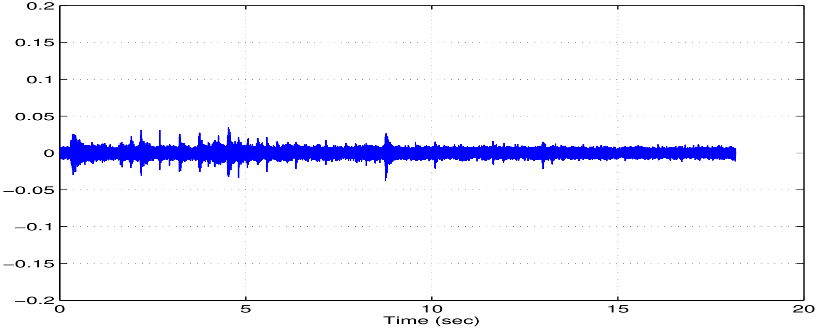


Figure: Diff. from the original: Aja: the smallest 700,791 DCT coefficients

This nonlinear approximation sounds *sharper* and *solid* than the linear one; and a way better than the standard basis!!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

  The boundary points of real signals are often neither periodic nor tied to 0 !!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

The boundary points of real signals are often neither periodic nor tied to 0 !!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

  The boundary points of real signals are often neither periodic nor tied to 0 !!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

  The boundary points of real signals are often neither periodic nor tied to 0 !!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

  The boundary points of real signals are often neither periodic nor tied to 0 !!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

The boundary points of real signals are often neither periodic nor tied to 0 !!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

The boundary points of real signals are often neither periodic nor tied to 0 !!

Remarks on DCT

- The basis vectors of the DCT have physical meaning, i.e., frequencies of their oscillations.
- Hence, the size of each expansion coefficient tells you the amount of the 'cosine waves' of the corresponding frequency in an input signal.
- As already mentioned, it is a fast transform with $O(n \log n)$ complexity.
- The *faster the decay* of the expansion coefficients is (as the frequency increases), the *smoother* an input signal is.
- The decay of the expansion coefficients w.r.t. DCT is generally faster than those using the Discrete Fourier Transform (DFT) and the Discrete Sine Transform (DST) due to the differences in the boundary conditions satisfied by these transforms:
  - DCT: the Neumann BC (the ends are free to move);
  - DFT: the periodic BC;
  - DST: the Dirichlet BC (the ends are tied to 0).

  The boundary points of real signals are often neither periodic nor tied to 0 !!

Outline

Data-Adaptive Bases

- The previous bases we considered (the standard basis, and the DCT basis) are *not* data adaptive.
- The (full) basis vectors are the same regardless of input vectors.
- The data adaptivity showed up only when we applied nonlinear approximation of the input vector.
- Now, we shall consider a truly data-adaptive basis, i.e., the basis vectors totally depends on an input vector.
- The set of basis vectors computed in Principal Component Analysis (PCA) (a.k.a. Karhunen-Loève Transform (KLT)) is one such example.

Data-Adaptive Bases

- The previous bases we considered (the standard basis, and the DCT basis) are *not* data adaptive.
- The (full) basis vectors are the same regardless of input vectors.
- The data adaptivity showed up only when we applied nonlinear approximation of the input vector.
- Now, we shall consider a truly data-adaptive basis, i.e., the basis vectors totally depends on an input vector.
- The set of basis vectors computed in Principal Component Analysis (PCA) (a.k.a. Karhunen-Loève Transform (KLT)) is one such example.

Data-Adaptive Bases

- The previous bases we considered (the standard basis, and the DCT basis) are *not* data adaptive.
- The (full) basis vectors are the same regardless of input vectors.
- The data adaptivity showed up only when we applied nonlinear approximation of the input vector.
- Now, we shall consider a truly data-adaptive basis, i.e., the basis vectors totally depends on an input vector.
- The set of basis vectors computed in Principal Component Analysis (PCA) (a.k.a. Karhunen-Loève Transform (KLT)) is one such example.

Data-Adaptive Bases

- The previous bases we considered (the standard basis, and the DCT basis) are *not* data adaptive.
- The (full) basis vectors are the same regardless of input vectors.
- The data adaptivity showed up only when we applied nonlinear approximation of the input vector.
- Now, we shall consider a truly data-adaptive basis, i.e., the basis vectors totally depends on an input vector.
- The set of basis vectors computed in Principal Component Analysis (PCA) (a.k.a. Karhunen-Loève Transform (KLT)) is one such example.

Data-Adaptive Bases

- The previous bases we considered (the standard basis, and the DCT basis) are *not* data adaptive.
- The (full) basis vectors are the same regardless of input vectors.
- The data adaptivity showed up only when we applied nonlinear approximation of the input vector.
- Now, we shall consider a truly data-adaptive basis, i.e., the basis vectors totally depends on an input vector.
- The set of basis vectors computed in Principal Component Analysis (PCA) (a.k.a. Karhunen-Loève Transform (KLT)) is one such example.

Basic Assumptions

- The starting point of PCA is to assume an underlying *stochastic process* $\boldsymbol{F} = (F_1, \ldots, F_n)^\mathsf{T} \in \mathbb{R}^n$ where each F_i is a *random variable* and $\boldsymbol{F}$ obeys a probability law described by the probability density function (pdf), say, $p_{\boldsymbol{F}}(f_1, \ldots, f_n)$.
- Observed data (or a set of input signals) are viewed as *realizations* of this stochastic process.

Basic Assumptions

- The starting point of PCA is to assume an underlying *stochastic process* $\boldsymbol{F} = (F_1, \ldots, F_n)^{\mathsf{T}} \in \mathbb{R}^n$ where each F_i is a *random variable* and $\boldsymbol{F}$ obeys a probability law described by the probability density function (pdf), say, $p_{\boldsymbol{F}}(f_1, \ldots, f_n)$.
- Observed data (or a set of input signals) are viewed as *realizations* of this stochastic process.

Basic Assumptions

- The starting point of PCA is to assume an underlying *stochastic process* $\boldsymbol{F} = (F_1, \ldots, F_n)^{\mathsf{T}} \in \mathbb{R}^n$ where each F_i is a *random variable* and $\boldsymbol{F}$ obeys a probability law described by the probability density function (pdf), say, $p_{\boldsymbol{F}}(f_1, \ldots, f_n)$.
- Observed data (or a set of input signals) are viewed as *realizations* of this stochastic process.

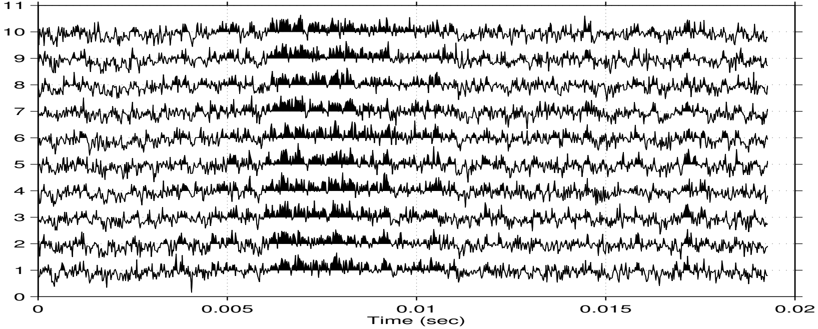


Figure: Ten realization of some stochastic process

Motivations

- Let $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\} \subset \mathbb{R}^n$ be the N realization (or observations) of the stochastic process $\boldsymbol{F}$.
- The case of $N > n$ is called the classical case while the case of $N < n$ is called the neoclassical case.
- A database of similar objects (e.g., a music database or a face image database) can be viewed as a collection of realizations from some (complicated) stochastic process.
- What is a good basis to represent and efficiently approximate such realizations *as a whole*?
- To answer that question, let us first consider the covariance ($\approx$ correlation) between ith and jth entries of the process $\boldsymbol{F}$:

$$\Gamma_{\boldsymbol{F}}(i,j) := \mathbb{E}\left[(F_i - \mathbb{E}F_i)(F_j - \mathbb{E}F_j)\right],$$

  where $\mathbb{E}$ is the *mathematical expectation*.
- We can put this in a matrix form

$$\Gamma_{\boldsymbol{F}} := \mathbb{E}\left[(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})^{\mathsf{T}}\right] = \mathbb{E}\left[\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}\right] - (\mathbb{E}\boldsymbol{F})(\mathbb{E}\boldsymbol{F})^{\mathsf{T}} \in \mathbb{R}^{n \times n}.$$

Motivations

- Let $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\} \subset \mathbb{R}^n$ be the N realization (or observations) of the stochastic process $\boldsymbol{F}$.
- The case of $N > n$ is called the classical case while the case of $N < n$ is called the neoclassical case.
- A database of similar objects (e.g., a music database or a face image database) can be viewed as a collection of realizations from some (complicated) stochastic process.
- What is a good basis to represent and efficiently approximate such realizations *as a whole*?
- To answer that question, let us first consider the covariance ($\approx$ correlation) between ith and jth entries of the process $\boldsymbol{F}$:

$$\Gamma_{\boldsymbol{F}}(i,j) := \mathbb{E}\left[(F_i - \mathbb{E}F_i)(F_j - \mathbb{E}F_j)\right],$$

  where $\mathbb{E}$ is the *mathematical expectation*.
- We can put this in a matrix form

$$\Gamma_{\boldsymbol{F}} := \mathbb{E}\left[(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})^{\mathsf{T}}\right] = \mathbb{E}\left[\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}\right] - (\mathbb{E}\boldsymbol{F})(\mathbb{E}\boldsymbol{F})^{\mathsf{T}} \in \mathbb{R}^{n\times n}.$$

Motivations

- Let $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\} \subset \mathbb{R}^n$ be the N realization (or observations) of the stochastic process $\boldsymbol{F}$.
- The case of $N > n$ is called the classical case while the case of $N < n$ is called the neoclassical case.
- A database of similar objects (e.g., a music database or a face image database) can be viewed as a collection of realizations from some (complicated) stochastic process.
- What is a good basis to represent and efficiently approximate such realizations *as a whole*?
- To answer that question, let us first consider the covariance ($\approx$ correlation) between ith and jth entries of the process $\boldsymbol{F}$:

$$\Gamma_{\boldsymbol{F}}(i,j) := \mathbb{E}\left[(F_i - \mathbb{E}F_i)(F_j - \mathbb{E}F_j)\right],$$

  where $\mathbb{E}$ is the *mathematical expectation*.
- We can put this in a matrix form

$$\Gamma_{\boldsymbol{F}} := \mathbb{E}\left[(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})^{\mathsf{T}}\right] = \mathbb{E}\left[\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}\right] - (\mathbb{E}\boldsymbol{F})(\mathbb{E}\boldsymbol{F})^{\mathsf{T}} \in \mathbb{R}^{n \times n}.$$

Motivations

- Let $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\} \subset \mathbb{R}^n$ be the N realization (or observations) of the stochastic process $\boldsymbol{F}$.
- The case of $N > n$ is called the classical case while the case of $N < n$ is called the neoclassical case.
- A database of similar objects (e.g., a music database or a face image database) can be viewed as a collection of realizations from some (complicated) stochastic process.
- What is a good basis to represent and efficiently approximate such realizations *as a whole*?
- To answer that question, let us first consider the covariance ($\approx$ correlation) between ith and jth entries of the process $\boldsymbol{F}$:

$$\Gamma_{\boldsymbol{F}}(i,j) := \mathbb{E}\left[(F_i - \mathbb{E}F_i)(F_j - \mathbb{E}F_j)\right],$$

where $\mathbb{E}$ is the *mathematical expectation*.
- We can put this in a matrix form

$$\Gamma_{\boldsymbol{F}} := \mathbb{E}\left[(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})^{\mathsf{T}}\right] = \mathbb{E}\left[\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}\right] - (\mathbb{E}\boldsymbol{F})(\mathbb{E}\boldsymbol{F})^{\mathsf{T}} \in \mathbb{R}^{n \times n}.$$

Motivations

- Let $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\} \subset \mathbb{R}^n$ be the N realization (or observations) of the stochastic process $\boldsymbol{F}$.
- The case of $N > n$ is called the classical case while the case of $N < n$ is called the neoclassical case.
- A database of similar objects (e.g., a music database or a face image database) can be viewed as a collection of realizations from some (complicated) stochastic process.
- What is a good basis to represent and efficiently approximate such realizations *as a whole*?
- To answer that question, let us first consider the covariance ($\approx$ correlation) between ith and jth entries of the process $\boldsymbol{F}$:

$$\Gamma_{\boldsymbol{F}}(i,j) := \mathbb{E}\left[(F_i - \mathbb{E}F_i)(F_j - \mathbb{E}F_j)\right],$$

  where $\mathbb{E}$ is the *mathematical expectation*.
- We can put this in a matrix form

$$\Gamma_{\boldsymbol{F}} := \mathbb{E}\left[(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})^{\mathsf{T}}\right] = \mathbb{E}\left[\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}\right] - (\mathbb{E}\boldsymbol{F})(\mathbb{E}\boldsymbol{F})^{\mathsf{T}} \in \mathbb{R}^{n \times n}.$$

Motivations

- Let $\{\boldsymbol{f}_1, \dots, \boldsymbol{f}_N\} \subset \mathbb{R}^n$ be the N realization (or observations) of the stochastic process $\boldsymbol{F}$.
- The case of $N > n$ is called the classical case while the case of $N < n$ is called the neoclassical case.
- A database of similar objects (e.g., a music database or a face image database) can be viewed as a collection of realizations from some (complicated) stochastic process.
- What is a good basis to represent and efficiently approximate such realizations *as a whole*?
- To answer that question, let us first consider the covariance ($\approx$ correlation) between ith and jth entries of the process $\boldsymbol{F}$:

$$\Gamma_{\boldsymbol{F}}(i, j) := \mathbb{E}\left[(F_i - \mathbb{E}F_i)(F_j - \mathbb{E}F_j)\right],$$

  where $\mathbb{E}$ is the *mathematical expectation*.
- We can put this in a matrix form

$$\Gamma_{\boldsymbol{F}} := \mathbb{E}\left[(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})(\boldsymbol{F} - \mathbb{E}\boldsymbol{F})^{\mathsf{T}}\right] = \mathbb{E}\left[\boldsymbol{F}\boldsymbol{F}^{\mathsf{T}}\right] - (\mathbb{E}\boldsymbol{F})(\mathbb{E}\boldsymbol{F})^{\mathsf{T}} \in \mathbb{R}^{n \times n}.$$

Derivation of PCA

- Since we are dealing with the N realizations instead of infinitely many realizations, we need to replace the above mathematical expectations by the so-called *sample estimates* that are computable using those N realizations.
- The sample estimate of $\Gamma_{\boldsymbol{F}}$ is:

$$\widehat{\Gamma}_{\boldsymbol{F}} := \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{f}_j \boldsymbol{f}_j^{\mathsf{T}} - \overline{\boldsymbol{f}}\, \overline{\boldsymbol{f}}^{\mathsf{T}},$$

  where $\overline{\boldsymbol{f}} := \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{f}_j$ is the *sample mean* of this process.
- Let's define the data matrix $F := [\boldsymbol{f}_1 | \cdots | \boldsymbol{f}_N] \in \mathbb{R}^{n \times N}$.
- Then, $\widehat{\Gamma}_{\boldsymbol{F}} = \frac{1}{N} F F^{\mathsf{T}} - \overline{\boldsymbol{f}}\, \overline{\boldsymbol{f}}^{\mathsf{T}}$.

Derivation of PCA

- Since we are dealing with the N realizations instead of infinitely many realizations, we need to replace the above mathematical expectations by the so-called *sample estimates* that are computable using those N realizations.
- The sample estimate of $\Gamma_{\boldsymbol{F}}$ is:

$$\widehat{\Gamma}_{\boldsymbol{F}} := \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{f}_j \boldsymbol{f}_j^{\mathsf{T}} - \overline{\boldsymbol{f}}\,\overline{\boldsymbol{f}}^{\mathsf{T}},$$

where $\overline{\boldsymbol{f}} := \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{f}_j$ is the *sample mean* of this process.

- Let's define the data matrix $F := [\boldsymbol{f}_1 | \cdots | \boldsymbol{f}_N] \in \mathbb{R}^{n \times N}$.
- Then, $\widehat{\Gamma}_{\boldsymbol{F}} = \frac{1}{N} F F^{\mathsf{T}} - \overline{\boldsymbol{f}}\,\overline{\boldsymbol{f}}^{\mathsf{T}}$.

Derivation of PCA

- Since we are dealing with the N realizations instead of infinitely many realizations, we need to replace the above mathematical expectations by the so-called *sample estimates* that are computable using those N realizations.
- The sample estimate of $\Gamma_{\boldsymbol{F}}$ is:

$$\widehat{\Gamma}_{\boldsymbol{F}} := \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{f}_j \boldsymbol{f}_j^{\mathsf{T}} - \overline{\boldsymbol{f}}\, \overline{\boldsymbol{f}}^{\mathsf{T}},$$

where $\overline{\boldsymbol{f}} := \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{f}_j$ is the *sample mean* of this process.

- Let's define the data matrix $F := \left[\boldsymbol{f}_1 | \cdots | \boldsymbol{f}_N\right] \in \mathbb{R}^{n \times N}$.
- Then, $\widehat{\Gamma}_{\boldsymbol{F}} = \frac{1}{N} F F^{\mathsf{T}} - \overline{\boldsymbol{f}}\, \overline{\boldsymbol{f}}^{\mathsf{T}}$.

Derivation of PCA

- Since we are dealing with the N realizations instead of infinitely many realizations, we need to replace the above mathematical expectations by the so-called *sample estimates* that are computable using those N realizations.
- The sample estimate of $\Gamma_{\boldsymbol{F}}$ is:

$$\widehat{\Gamma}_{\boldsymbol{F}} := \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{f}_j \boldsymbol{f}_j^{\mathsf{T}} - \overline{\boldsymbol{f}}\,\overline{\boldsymbol{f}}^{\mathsf{T}},$$

  where $\overline{\boldsymbol{f}} := \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{f}_j$ is the *sample mean* of this process.
- Let's define the data matrix $F := [\boldsymbol{f}_1 | \cdots | \boldsymbol{f}_N] \in \mathbb{R}^{n\times N}$.
- Then, $\widehat{\Gamma}_{\boldsymbol{F}} = \frac{1}{N} F F^{\mathsf{T}} - \overline{\boldsymbol{f}}\,\overline{\boldsymbol{f}}^{\mathsf{T}}$.

Derivation of PCA . . .

- Suppose we want to find a *data-adaptive* orthonormal basis of $\mathbb{R}^n$ such that the realizations of the process $\boldsymbol{F}$ as a whole (i.e., *on average*) can be best approximated by $n'(\ll n)$ coordinates in the sense of the mean-squared error.
- Let $W = [\boldsymbol{w}_1|\cdots|\boldsymbol{w}_n]$ be an orthogonal matrix in $\mathbb{R}^n$ and let $\boldsymbol{G} = W^{\mathsf{T}}\boldsymbol{F}$, i.e.,

$$\boldsymbol{F} = W\boldsymbol{G} = G_1\boldsymbol{w}_1 + \cdots + G_n\boldsymbol{w}_n.$$

- Suppose we retain $\{G_1, \ldots, G_{n'}\}$ and replace $\{G_{n'+1}, \ldots, G_n\}$ by some predetermined constants $\{\alpha_{n'+1}, \ldots, \alpha_n\}$, and view

$$\boldsymbol{F}^{(n')} := \sum_{k=1}^{n'} G_k \boldsymbol{w}_k + \sum_{k=n'+1}^{n} \alpha_k \boldsymbol{w}_k,$$

as an approximation to $\boldsymbol{F}$.
- The error of this approximation is of course

$$\Delta\boldsymbol{F} := \boldsymbol{F} - \boldsymbol{F}^{(n')} = \sum_{k=n'+1}^{n} (G_k - \alpha_k)\boldsymbol{w}_k$$

Derivation of PCA ...

- Suppose we want to find a *data-adaptive* orthonormal basis of $\mathbb{R}^n$ such that the realizations of the process $\boldsymbol{F}$ as a whole (i.e., *on average*) can be best approximated by $n'(\ll n)$ coordinates in the sense of the mean-squared error.
- Let $W = [\boldsymbol{w}_1|\cdots|\boldsymbol{w}_n]$ be an orthogonal matrix in $\mathbb{R}^n$ and let $\boldsymbol{G} = W^\mathsf{T}\boldsymbol{F}$, i.e.,

$$\boldsymbol{F} = W\boldsymbol{G} = G_1\boldsymbol{w}_1 + \cdots + G_n\boldsymbol{w}_n.$$

- Suppose we retain $\{G_1, \ldots, G_{n'}\}$ and replace $\{G_{n'+1}, \ldots, G_n\}$ by some predetermined constants $\{\alpha_{n'+1}, \ldots, \alpha_n\}$, and view

$$\boldsymbol{F}^{(n')} := \sum_{k=1}^{n'} G_k\boldsymbol{w}_k + \sum_{k=n'+1}^{n} \alpha_k\boldsymbol{w}_k,$$

as an approximation to $\boldsymbol{F}$.
- The error of this approximation is of course

$$\Delta\boldsymbol{F} := \boldsymbol{F} - \boldsymbol{F}^{(n')} = \sum_{k=n'+1}^{n} (G_k - \alpha_k)\boldsymbol{w}_k$$

Derivation of PCA . . .

- Suppose we want to find a *data-adaptive* orthonormal basis of $\mathbb{R}^n$ such that the realizations of the process $\boldsymbol{F}$ as a whole (i.e., *on average*) can be best approximated by $n'(\ll n)$ coordinates in the sense of the mean-squared error.
- Let $W = [\boldsymbol{w}_1|\cdots|\boldsymbol{w}_n]$ be an orthogonal matrix in $\mathbb{R}^n$ and let $\boldsymbol{G} = W^{\mathsf{T}}\boldsymbol{F}$, i.e.,

$$\boldsymbol{F} = W\boldsymbol{G} = G_1\boldsymbol{w}_1 + \cdots + G_n\boldsymbol{w}_n.$$

- Suppose we retain $\{G_1, \ldots, G_{n'}\}$ and replace $\{G_{n'+1}, \ldots, G_n\}$ by some predetermined constants $\{\alpha_{n'+1}, \ldots, \alpha_n\}$, and view

$$\boldsymbol{F}^{(n')} := \sum_{k=1}^{n'} G_k\boldsymbol{w}_k + \sum_{k=n'+1}^{n} \alpha_k\boldsymbol{w}_k,$$

as an approximation to $\boldsymbol{F}$.
- The error of this approximation is of course

$$\Delta\boldsymbol{F} := \boldsymbol{F} - \boldsymbol{F}^{(n')} = \sum_{k=n'+1}^{n} (G_k - \alpha_k)\boldsymbol{w}_k$$

Derivation of PCA ...

- Suppose we want to find a *data-adaptive* orthonormal basis of $\mathbb{R}^n$ such that the realizations of the process $\boldsymbol{F}$ as a whole (i.e., *on average*) can be best approximated by $n'(\ll n)$ coordinates in the sense of the mean-squared error.
- Let $W = [\boldsymbol{w}_1|\cdots|\boldsymbol{w}_n]$ be an orthogonal matrix in $\mathbb{R}^n$ and let $\boldsymbol{G} = W^{\mathsf{T}}\boldsymbol{F}$, i.e.,
$$\boldsymbol{F} = W\boldsymbol{G} = G_1\boldsymbol{w}_1 + \cdots + G_n\boldsymbol{w}_n.$$
- Suppose we retain $\{G_1,\ldots,G_{n'}\}$ and replace $\{G_{n'+1},\ldots,G_n\}$ by some predetermined constants $\{\alpha_{n'+1},\ldots,\alpha_n\}$, and view
$$\boldsymbol{F}^{(n')} := \sum_{k=1}^{n'} G_k\boldsymbol{w}_k + \sum_{k=n'+1}^{n} \alpha_k\boldsymbol{w}_k,$$
as an approximation to $\boldsymbol{F}$.
- The error of this approximation is of course
$$\Delta\boldsymbol{F} := \boldsymbol{F} - \boldsymbol{F}^{(n')} = \sum_{k=n'+1}^{n} (G_k - \alpha_k)\boldsymbol{w}_k$$

Derivation of PCA . . .

- Hence, the mean-squared error is:

$$
\begin{aligned}
\mathrm{Err}(\alpha_{n'+1},\dots,\alpha_n) &:= \mathbb{E}\left[\|\Delta \boldsymbol{F}\|^2\right] \quad \text{where } \|\cdot\| \text{ is the 2-norm in } \mathbb{R}^n \\
&= \mathbb{E}\left[(\Delta \boldsymbol{F})^{\mathsf{T}} \Delta \boldsymbol{F}\right] \\
&= \mathbb{E}\left[\sum_{k=n'+1}^{n} \sum_{l=n'+1}^{n} (G_k-\alpha_k)(G_l-\alpha_l)\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{w}_l\right] \\
&= \mathbb{E}\left[\sum_{k=n'+1}^{n} (G_k-\alpha_k)^2\right] \quad \text{since } \boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{w}_l = \delta_{kl}.
\end{aligned}
$$

- We want to find $\{\alpha_k\}_{k=n'+1}^{n}$ that *minimize* $\mathrm{Err}(\alpha_{n'+1},\dots,\alpha_n)$.
- Let $\{\alpha^*_{n'+1},\dots,\alpha^*_n\}$ be the minimizer of $\mathrm{Err}(\alpha_{n'+1},\dots,\alpha_n)$. Then, α^*_k should satisfy

$$
\frac{\partial \mathrm{Err}(\alpha_{n'+1},\dots,\alpha_n)}{\partial \alpha_k} = -2\mathbb{E}(G_k-\alpha_k) = 0,
$$

which easily leads to $\alpha^*_k = \mathbb{E}[G_k] = \mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{F}\right]$, $k = n'+1,\dots,n$.

Derivation of PCA . . .

- Hence, the mean-squared error is:

$$\begin{aligned}
\mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n) &:= \mathbb{E}\left[\|\Delta\boldsymbol{F}\|^2\right] \quad \text{where } \|\cdot\| \text{ is the 2-norm in } \mathbb{R}^n \\
&= \mathbb{E}\left[(\Delta\boldsymbol{F})^{\mathsf{T}}\Delta\boldsymbol{F}\right] \\
&= \mathbb{E}\left[\sum_{k=n'+1}^{n}\sum_{l=n'+1}^{n}(G_k-\alpha_k)(G_l-\alpha_l)\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{w}_l\right] \\
&= \mathbb{E}\left[\sum_{k=n'+1}^{n}(G_k-\alpha_k)^2\right] \quad \text{since } \boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{w}_l = \delta_{kl}.
\end{aligned}$$

- We want to find $\{\alpha_k\}_{k=n'+1}^{n}$ that *minimize* $\mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n)$.
- Let $\{\alpha^*_{n'+1},\ldots,\alpha^*_n\}$ be the minimizer of $\mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n)$. Then, α^*_k should satisfy

$$\frac{\partial \mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n)}{\partial \alpha_k} = -2\mathbb{E}(G_k-\alpha_k) = 0,$$

which easily leads to $\alpha^*_k = \mathbb{E}[G_k] = \mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{F}\right]$, $k = n'+1,\ldots,n$.

Derivation of PCA ...

- Hence, the mean-squared error is:

$$\begin{aligned}
\mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n) &:= \mathbb{E}\left[\|\Delta \boldsymbol{F}\|^2\right] \quad \text{where } \|\cdot\| \text{ is the 2-norm in } \mathbb{R}^n \\
&= \mathbb{E}\left[(\Delta \boldsymbol{F})^{\mathsf{T}} \Delta \boldsymbol{F}\right] \\
&= \mathbb{E}\left[\sum_{k=n'+1}^{n} \sum_{l=n'+1}^{n} (G_k - \alpha_k)(G_l - \alpha_l) \boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{w}_l\right] \\
&= \mathbb{E}\left[\sum_{k=n'+1}^{n} (G_k - \alpha_k)^2\right] \quad \text{since } \boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{w}_l = \delta_{kl}.
\end{aligned}$$

- We want to find $\{\alpha_k\}_{k=n'+1}^{n}$ that *minimize* $\mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n)$.
- Let $\{\alpha^*_{n'+1},\ldots,\alpha^*_n\}$ be the minimizer of $\mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n)$. Then, α^*_k should satisfy

$$\frac{\partial \mathrm{Err}(\alpha_{n'+1},\ldots,\alpha_n)}{\partial \alpha_k} = -2\mathbb{E}(G_k - \alpha_k) = 0,$$

which easily leads to $\alpha^*_k = \mathbb{E}[G_k] = \mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{F}\right]$, $k = n'+1,\ldots,n$.

Derivation of PCA ...

Hence, we have:

$$
\begin{aligned}
\mathrm{Err}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right) &= \mathbb{E}\left[\sum_{k=n'+1}^{n}(G_k-\alpha^*_k)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[(G_k-\mathbb{E}[G_k])^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^{\mathsf{T}}\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\,\boldsymbol{w}_k\right] \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\mathbb{E}\left[(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\right]\boldsymbol{w}_k \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k
\end{aligned}
$$

Derivation of PCA . . .

Hence, we have:

$$
\begin{aligned}
\mathrm{Err}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right) &= \mathbb{E}\left[\sum_{k=n'+1}^{n}(G_k-\alpha^*_k)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[(G_k-\mathbb{E}[G_k])^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^{\mathsf{T}}\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\boldsymbol{w}_k\right] \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\mathbb{E}\left[(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\right]\boldsymbol{w}_k \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k
\end{aligned}
$$

Derivation of PCA ...

Hence, we have:

$$
\begin{aligned}
\mathrm{Err}\left(\alpha^*_{n'+1},\dots,\alpha^*_n\right) &= \mathbb{E}\left[\sum_{k=n'+1}^{n}(G_k-\alpha^*_k)^2\right] \\
&= \sum_{k=n'+1}^{n} \mathbb{E}\left[(G_k-\mathbb{E}[G_k])^2\right] \\
&= \sum_{k=n'+1}^{n} \mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^2\right] \\
&= \sum_{k=n'+1}^{n} \mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^{\mathsf{T}}\right] \\
&= \sum_{k=n'+1}^{n} \mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\boldsymbol{w}_k\right] \\
&= \sum_{k=n'+1}^{n} \boldsymbol{w}_k^{\mathsf{T}}\mathbb{E}\left[(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\right]\boldsymbol{w}_k \\
&= \sum_{k=n'+1}^{n} \boldsymbol{w}_k^{\mathsf{T}}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k
\end{aligned}
$$

Derivation of PCA . . .

Hence, we have:

$$\begin{aligned}
\mathrm{Err}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right) &= \mathbb{E}\left[\sum_{k=n'+1}^{n}(G_k-\alpha^*_k)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[(G_k-\mathbb{E}[G_k])^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^{\mathsf{T}}\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\,\boldsymbol{w}_k\right] \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\mathbb{E}\left[(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\right]\boldsymbol{w}_k \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k
\end{aligned}$$

Derivation of PCA ...

Hence, we have:

$$\begin{aligned}
\mathrm{Err}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right) &= \mathbb{E}\left[\sum_{k=n'+1}^{n}(G_k-\alpha^*_k)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[(G_k-\mathbb{E}[G_k])^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^{\mathsf{T}}\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\,\boldsymbol{w}_k\right] \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\mathbb{E}\left[(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\right]\boldsymbol{w}_k \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k
\end{aligned}$$

Derivation of PCA ...

Hence, we have:

$$
\begin{aligned}
\mathrm{Err}\left(\alpha^*_{n'+1},\dots,\alpha^*_n\right) &= \mathbb{E}\left[\sum_{k=n'+1}^{n}(G_k-\alpha^*_k)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[(G_k-\mathbb{E}[G_k])^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^2\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)\left(\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\right)^{\mathsf{T}}\right] \\
&= \sum_{k=n'+1}^{n}\mathbb{E}\left[\boldsymbol{w}_k^{\mathsf{T}}(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\,\boldsymbol{w}_k\right] \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\mathbb{E}\left[(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F}-\mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\right]\boldsymbol{w}_k \\
&= \sum_{k=n'+1}^{n}\boldsymbol{w}_k^{\mathsf{T}}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k
\end{aligned}
$$

Derivation of PCA ...

- Finally, let's find $\{\boldsymbol{w}_k\}_{k=n'+1}^{n} \subset \mathbb{R}^n$ that minimize the above $\mathrm{Err}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right)$ subject to $\boldsymbol{w}_k^\mathsf{T}\boldsymbol{w}_k = 1,\ k = n'+1, \ldots, n$.
- To do so, we use the *Lagrange multipliers*, $\{\lambda_{n'+1}, \ldots, \lambda_n\} \subset \mathbb{R}$:

$$\begin{aligned}\widetilde{\mathrm{Err}}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right) &:= \mathrm{Err}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right) - \sum_{k=n'+1}^{n} \lambda_k \left(\boldsymbol{w}_k^\mathsf{T}\boldsymbol{w}_k - 1\right) \\ &= \sum_{k=n'+1}^{n} \left\{\boldsymbol{w}_k^\mathsf{T}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k - \lambda_k\left(\boldsymbol{w}_k^\mathsf{T}\boldsymbol{w}_k - 1\right)\right\}.\end{aligned}$$

- $\boldsymbol{w}_k$ minimizing the above should satisfy:

$$\frac{\partial \widetilde{\mathrm{Err}}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right)}{\partial \boldsymbol{w}_k} = 2\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k - 2\lambda_k\boldsymbol{w}_k = 0,$$

which leads to the following eigenvalue problem:

$$\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k = \lambda_k\boldsymbol{w}_k, \quad k = n'+1, \ldots, n.$$

Derivation of PCA ...

- Finally, let's find $\{\boldsymbol{w}_k\}_{k=n'+1}^{n} \subset \mathbb{R}^n$ that minimize the above $\mathrm{Err}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right)$ subject to $\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{w}_k = 1,\ k = n'+1,\ldots,n$.
- To do so, we use the *Lagrange multipliers*, $\{\lambda_{n'+1},\ldots,\lambda_n\} \subset \mathbb{R}$:

$$\begin{aligned}\widetilde{\mathrm{Err}}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right) &:= \mathrm{Err}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right) - \sum_{k=n'+1}^{n} \lambda_k \left(\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{w}_k - 1\right) \\ &= \sum_{k=n'+1}^{n} \left\{\boldsymbol{w}_k^{\mathsf{T}}\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k - \lambda_k\left(\boldsymbol{w}_k^{\mathsf{T}}\boldsymbol{w}_k - 1\right)\right\}.\end{aligned}$$

- $\boldsymbol{w}_k$ minimizing the above should satisfy:

$$\frac{\partial \widetilde{\mathrm{Err}}\left(\alpha^*_{n'+1},\ldots,\alpha^*_n\right)}{\partial \boldsymbol{w}_k} = 2\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k - 2\lambda_k\boldsymbol{w}_k = 0,$$

which leads to the following eigenvalue problem:

$$\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k = \lambda_k\boldsymbol{w}_k, \quad k = n'+1,\ldots,n.$$

Derivation of PCA ...

- Finally, let's find $\{\boldsymbol{w}_k\}_{k=n'+1}^{n} \subset \mathbb{R}^n$ that minimize the above $\mathrm{Err}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right)$ subject to $\boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{w}_k = 1,\ k = n'+1, \ldots, n$.
- To do so, we use the *Lagrange multipliers*, $\{\lambda_{n'+1}, \ldots, \lambda_n\} \subset \mathbb{R}$:

$$\begin{aligned}\widetilde{\mathrm{Err}}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right) &:= \mathrm{Err}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right) - \sum_{k=n'+1}^{n} \lambda_k \left(\boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{w}_k - 1\right) \\ &= \sum_{k=n'+1}^{n} \left\{\boldsymbol{w}_k^{\mathsf{T}} \Gamma_{\boldsymbol{F}} \boldsymbol{w}_k - \lambda_k \left(\boldsymbol{w}_k^{\mathsf{T}} \boldsymbol{w}_k - 1\right)\right\}.\end{aligned}$$

- $\boldsymbol{w}_k$ minimizing the above should satisfy:

$$\frac{\partial \widetilde{\mathrm{Err}}\left(\alpha_{n'+1}^*, \ldots, \alpha_n^*\right)}{\partial \boldsymbol{w}_k} = 2\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k - 2\lambda_k \boldsymbol{w}_k = 0,$$

which leads to the following eigenvalue problem:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \quad k = n'+1, \ldots, n.$$

Derivation of PCA . . .

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n] \in \mathbb{R}^{n \times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \; k = 1, \ldots, n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W \Lambda, \; \Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

- Let $W_{\mathrm{PCA}} \in \mathbb{R}^{n \times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\mathrm{PCA}}^{\mathsf{T}} \boldsymbol{F} =: \boldsymbol{F}_{\mathrm{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned} \Gamma_{\boldsymbol{F}_{\mathrm{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])^{\mathsf{T}}\right] \\ &= \mathbb{E}\left[W_{\mathrm{PCA}}^{\mathsf{T}} (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])(\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}} W_{\mathrm{PCA}}\right] \\ &= W_{\mathrm{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\mathrm{PCA}}}_{= W_{\mathrm{PCA}} \Lambda} = \underbrace{W_{\mathrm{PCA}}^{\mathsf{T}} W_{\mathrm{PCA}}}_{= I_n} \Lambda = \Lambda, \end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\mathrm{PCA},i} - \mathbb{E}F_{\mathrm{PCA},i})(F_{\mathrm{PCA},j} - \mathbb{E}F_{\mathrm{PCA},j})\right] = \lambda_i \delta_{ij}$.

Derivation of PCA ...

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n] \in \mathbb{R}^{n \times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \ k = 1, \ldots, n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W \Lambda, \ \Lambda := \mathrm{diag}(\lambda_1, \ldots, \lambda_n).$$

- Let $W_{\mathrm{PCA}} \in \mathbb{R}^{n \times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\mathrm{PCA}}^{\mathsf{T}} \boldsymbol{F} =: \boldsymbol{F}_{\mathrm{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned}
\Gamma_{\boldsymbol{F}_{\mathrm{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}]) (\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])^{\mathsf{T}} \right] \\
&= \mathbb{E}\left[W_{\mathrm{PCA}}^{\mathsf{T}} (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}]) (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}} W_{\mathrm{PCA}} \right] \\
&= W_{\mathrm{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\mathrm{PCA}}}_{= W_{\mathrm{PCA}} \Lambda} = \underbrace{W_{\mathrm{PCA}}^{\mathsf{T}} W_{\mathrm{PCA}}}_{= I_n} \Lambda = \Lambda,
\end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\mathrm{PCA},i} - \mathbb{E} F_{\mathrm{PCA},i})(F_{\mathrm{PCA},j} - \mathbb{E} F_{\mathrm{PCA},j}) \right] = \lambda_i \delta_{ij}$.

Derivation of PCA ...

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1|\cdots|\boldsymbol{w}_n] \in \mathbb{R}^{n\times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k,\ k = 1,\dots,n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W\Lambda,\ \Lambda := \mathrm{diag}(\lambda_1,\dots,\lambda_n).$$

- Let $W_{\mathrm{PCA}} \in \mathbb{R}^{n\times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\mathrm{PCA}}^{\mathsf{T}}\boldsymbol{F} =: \boldsymbol{F}_{\mathrm{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned}
\Gamma_{\boldsymbol{F}_{\mathrm{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])^{\mathsf{T}}\right] \\
&= \mathbb{E}\left[W_{\mathrm{PCA}}^{\mathsf{T}}(\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])(\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}} W_{\mathrm{PCA}}\right] \\
&= W_{\mathrm{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\mathrm{PCA}}}_{=W_{\mathrm{PCA}}\Lambda} = \underbrace{W_{\mathrm{PCA}}^{\mathsf{T}} W_{\mathrm{PCA}}}_{=I_n}\Lambda = \Lambda,
\end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\mathrm{PCA},i} - \mathbb{E}F_{\mathrm{PCA},i})(F_{\mathrm{PCA},j} - \mathbb{E}F_{\mathrm{PCA},j})\right] = \lambda_i \delta_{ij}$.

Derivation of PCA . . .

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n] \in \mathbb{R}^{n \times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \; k = 1, \ldots, n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W\Lambda, \; \Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

- Let $W_{\mathrm{PCA}} \in \mathbb{R}^{n \times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\mathrm{PCA}}^{\mathsf{T}} \boldsymbol{F} =: \boldsymbol{F}_{\mathrm{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned}
\Gamma_{\boldsymbol{F}_{\mathrm{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])\,(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])^{\mathsf{T}}\right] \\
&= \mathbb{E}\left[W_{\mathrm{PCA}}^{\mathsf{T}}\,(\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])\,(\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}}\, W_{\mathrm{PCA}}\right] \\
&= W_{\mathrm{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\mathrm{PCA}}}_{= W_{\mathrm{PCA}}\Lambda} = \underbrace{W_{\mathrm{PCA}}^{\mathsf{T}} W_{\mathrm{PCA}}}_{= I_n} \Lambda = \Lambda,
\end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\mathrm{PCA},i} - \mathbb{E}F_{\mathrm{PCA},i})(F_{\mathrm{PCA},j} - \mathbb{E}F_{\mathrm{PCA},j})\right] = \lambda_i \delta_{ij}$.

Derivation of PCA . . .

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n] \in \mathbb{R}^{n \times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \ k = 1, \ldots, n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W \Lambda, \ \Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

- Let $W_{\text{PCA}} \in \mathbb{R}^{n \times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\text{PCA}}^{\mathsf{T}} \boldsymbol{F} =: \boldsymbol{F}_{\text{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned} \Gamma_{\boldsymbol{F}_{\text{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\text{PCA}} - \mathbb{E}[\boldsymbol{F}_{\text{PCA}}]) (\boldsymbol{F}_{\text{PCA}} - \mathbb{E}[\boldsymbol{F}_{\text{PCA}}])^{\mathsf{T}} \right] \\ &= \mathbb{E}\left[W_{\text{PCA}}^{\mathsf{T}} (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}]) (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}} W_{\text{PCA}} \right] \\ &= W_{\text{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\text{PCA}}}_{= W_{\text{PCA}} \Lambda} = \underbrace{W_{\text{PCA}}^{\mathsf{T}} W_{\text{PCA}}}_{= I_n} \Lambda = \Lambda, \end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\text{PCA},i} - \mathbb{E} F_{\text{PCA},i})(F_{\text{PCA},j} - \mathbb{E} F_{\text{PCA},j}) \right] = \lambda_i \delta_{ij}$.

Derivation of PCA ...

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1|\cdots|\boldsymbol{w}_n] \in \mathbb{R}^{n\times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \; k = 1, \ldots, n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W\Lambda, \; \Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

- Let $W_{\mathrm{PCA}} \in \mathbb{R}^{n\times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\mathrm{PCA}}^{\mathsf{T}} \boldsymbol{F} =: \boldsymbol{F}_{\mathrm{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned}
\Gamma_{\boldsymbol{F}_{\mathrm{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])^{\mathsf{T}}\right] \\
&= \mathbb{E}\left[W_{\mathrm{PCA}}^{\mathsf{T}} (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])(\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}} W_{\mathrm{PCA}}\right] \\
&= W_{\mathrm{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\mathrm{PCA}}}_{= W_{\mathrm{PCA}}\Lambda} = \underbrace{W_{\mathrm{PCA}}^{\mathsf{T}} W_{\mathrm{PCA}}}_{= I_n} \Lambda = \Lambda,
\end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\mathrm{PCA},i} - \mathbb{E}F_{\mathrm{PCA},i})(F_{\mathrm{PCA},j} - \mathbb{E}F_{\mathrm{PCA},j})\right] = \lambda_i \delta_{ij}$.

Derivation of PCA . . .

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n] \in \mathbb{R}^{n \times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \ k = 1, \ldots, n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W \Lambda, \ \Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

- Let $W_{\mathrm{PCA}} \in \mathbb{R}^{n \times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\mathrm{PCA}}^{\mathsf{T}} \boldsymbol{F} =: \boldsymbol{F}_{\mathrm{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned}
\Gamma_{\boldsymbol{F}_{\mathrm{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}]) (\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])^{\mathsf{T}} \right] \\
&= \mathbb{E}\left[W_{\mathrm{PCA}}^{\mathsf{T}} (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}]) (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}} W_{\mathrm{PCA}} \right] \\
&= W_{\mathrm{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\mathrm{PCA}}}_{= W_{\mathrm{PCA}} \Lambda} = \underbrace{W_{\mathrm{PCA}}^{\mathsf{T}} W_{\mathrm{PCA}}}_{= I_n} \Lambda = \Lambda,
\end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\mathrm{PCA},i} - \mathbb{E} F_{\mathrm{PCA},i})(F_{\mathrm{PCA},j} - \mathbb{E} F_{\mathrm{PCA},j}) \right] = \lambda_i \delta_{ij}$.

Derivation of PCA . . .

- Since n' was arbitrary as long as $1 \le n' \le n$, $W = [\boldsymbol{w}_1 | \cdots | \boldsymbol{w}_n] \in \mathbb{R}^{n \times n}$ the best basis matrix in the mean-squared error sense should satisfy:

$$\Gamma_{\boldsymbol{F}} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \; k = 1, \ldots, n, \text{ i.e., } \Gamma_{\boldsymbol{F}} W = W \Lambda, \; \Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

- Let $W_{\mathrm{PCA}} \in \mathbb{R}^{n \times n}$ be the above eigenvector matrix W. Analyzing the input process $\boldsymbol{F}$ not in the standard basis but in the eigenvector basis is called the Principal Component Analysis. The transformed process $\boldsymbol{G} = W_{\mathrm{PCA}}^{\mathsf{T}} \boldsymbol{F} =: \boldsymbol{F}_{\mathrm{PCA}}$ are called the Principal Componenets of $\boldsymbol{F}$.
- PCA provides the decorrelated coordinates as follows:

$$\begin{aligned}
\Gamma_{\boldsymbol{F}_{\mathrm{PCA}}} &= \mathbb{E}\left[(\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}]) (\boldsymbol{F}_{\mathrm{PCA}} - \mathbb{E}[\boldsymbol{F}_{\mathrm{PCA}}])^{\mathsf{T}} \right] \\
&= \mathbb{E}\left[W_{\mathrm{PCA}}^{\mathsf{T}} (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}]) (\boldsymbol{F} - \mathbb{E}[\boldsymbol{F}])^{\mathsf{T}} W_{\mathrm{PCA}} \right] \\
&= W_{\mathrm{PCA}}^{\mathsf{T}} \underbrace{\Gamma_{\boldsymbol{F}} W_{\mathrm{PCA}}}_{= W_{\mathrm{PCA}} \Lambda} = \underbrace{W_{\mathrm{PCA}}^{\mathsf{T}} W_{\mathrm{PCA}}}_{= I_n} \Lambda = \Lambda,
\end{aligned}$$

i.e., $\mathbb{E}\left[(F_{\mathrm{PCA},i} - \mathbb{E} F_{\mathrm{PCA},i})(F_{\mathrm{PCA},j} - \mathbb{E} F_{\mathrm{PCA},j}) \right] = \lambda_i \delta_{ij}$.

Remarks on PCA

- PCA is also known as the Karhunen-Loève Transform (KLT).
- In practice, the above covariance matrix $\Gamma_{\boldsymbol{F}}$ must be replaced by the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ based on the available observations $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$.
- Note that in the *classical setting* of $n \ll N$ (e.g., the census), the quality of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ is good whereas in the *neoclassical setting* of $n \gg N$ (e.g., image databases), that quality is poor.
- This implies that under the neoclassical setting, the PCA may not be effective; in fact, only the first $N-1$ basis vectors are meaningful.
- In practice, it would be best to use the Singular Value Decomposition (SVD) of the data matrix F instead of using the eigenvalue decomposition of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$.

Remarks on PCA

- PCA is also known as the Karhunen-Loève Transform (KLT).
- In practice, the above covariance matrix $\Gamma_{\boldsymbol{F}}$ must be replaced by the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ based on the available observations $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$.
- Note that in the *classical setting* of $n \ll N$ (e.g., the census), the quality of the sample covariance matrix $\widehat{\Gamma}_F$ is good whereas in the *neoclassical setting* of $n \gg N$ (e.g., image databases), that quality is poor.
- This implies that under the neoclassical setting, the PCA may not be effective; in fact, only the first $N-1$ basis vectors are meaningful.
- In practice, it would be best to use the Singular Value Decomposition (SVD) of the data matrix F instead of using the eigenvalue decomposition of the sample covariance matrix $\widehat{\Gamma}_F$.

Remarks on PCA

- PCA is also known as the Karhunen-Loève Transform (KLT).
- In practice, the above covariance matrix $\Gamma_{\boldsymbol{F}}$ must be replaced by the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ based on the available observations $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$.
- Note that in the *classical setting* of $n \ll N$ (e.g., the census), the quality of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ is good whereas in the *neoclassical setting* of $n \gg N$ (e.g., image databases), that quality is poor.
- This implies that under the neoclassical setting, the PCA may not be effective; in fact, only the first $N-1$ basis vectors are meaningful.
- In practice, it would be best to use the Singular Value Decomposition (SVD) of the data matrix F instead of using the eigenvalue decomposition of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$.

Remarks on PCA

- PCA is also known as the Karhunen-Loève Transform (KLT).
- In practice, the above covariance matrix $\Gamma_{\boldsymbol{F}}$ must be replaced by the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ based on the available observations $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$.
- Note that in the *classical setting* of $n \ll N$ (e.g., the census), the quality of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ is good whereas in the *neoclassical setting* of $n \gg N$ (e.g., image databases), that quality is poor.
- This implies that under the neoclassical setting, the PCA may not be effective; in fact, only the first $N-1$ basis vectors are meaningful.
- In practice, it would be best to use the Singular Value Decomposition (SVD) of the data matrix F instead of using the eigenvalue decomposition of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$.

Remarks on PCA

- PCA is also known as the Karhunen-Loève Transform (KLT).
- In practice, the above covariance matrix $\Gamma_{\boldsymbol{F}}$ must be replaced by the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ based on the available observations $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$.
- Note that in the *classical setting* of $n \ll N$ (e.g., the census), the quality of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$ is good whereas in the *neoclassical setting* of $n \gg N$ (e.g., image databases), that quality is poor.
- This implies that under the neoclassical setting, the PCA may not be effective; in fact, only the first $N-1$ basis vectors are meaningful.
- In practice, it would be best to use the Singular Value Decomposition (SVD) of the data matrix F instead of using the eigenvalue decomposition of the sample covariance matrix $\widehat{\Gamma}_{\boldsymbol{F}}$.

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $F \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $\boldsymbol{F} \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $\boldsymbol{F} \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $\boldsymbol{F} \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $\boldsymbol{F} \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $\boldsymbol{F} \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $\boldsymbol{F} \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

An Example PCA using the Music Signal

- Let's return to our music signal $\boldsymbol{f}$ with $n = 800,791$ samples.
- If we view this whole signal as a single realization $N = 1$ of the stochastic process $\boldsymbol{F} \in \mathbb{R}^n$, then the sample mean and covariance matrix are meaningless since the former is the signal itself and the latter is 0 matrix.
- Hence, one possibility is to *chop* this signal into $N \gg 1$ equal segments each of which contains n/N samples. Then consider these N vectors as the N realizations of the process $\boldsymbol{F} \in \mathbb{R}^{n/N}$.
- Let's take $N = 941$. Then $n/N = 851 =: m$.
- We can increase $N > 941$ if we allow the segments to overlap.
- For example, we can randomly select $N = 10,000$ starting locations in the index range between 1 and $799,941 (= 800,791 - 850)$.
- A big question remains: is it reasonable to view these N segments as N realizations of a single stochastic process $\boldsymbol{F} \in \mathbb{R}^m$?
- Nonetheless, let's proceed!

Ten Segments of the Music Signal

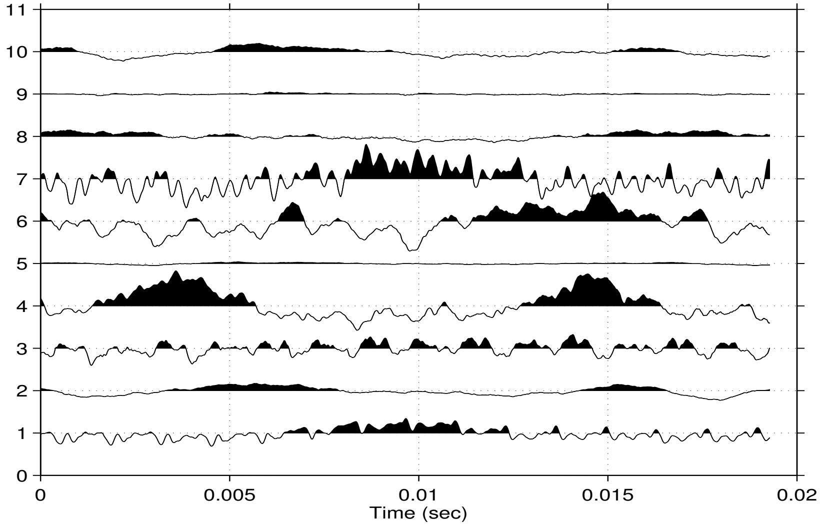


Figure: Each segment starts at a random location and contains $m = 851$ samples

PCA-based Approximation of the Music Signal

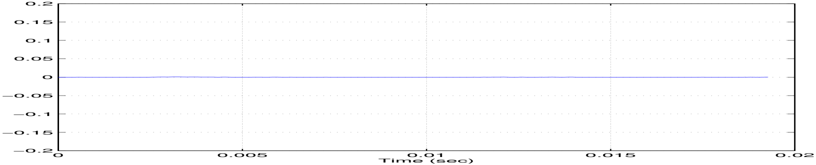
- Randomly pick the $N = 10,000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{f} \in \mathbb{R}^m$.
- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{f}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800,791$ principal components.
- Do the linear and nonlinear approximations as before.

PCA-based Approximation of the Music Signal

- Randomly pick the $N = 10,000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{\boldsymbol{f}} \in \mathbb{R}^m$.
- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{\boldsymbol{f}}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800,791$ principal components.
- Do the linear and nonlinear approximations as before.

PCA-based Approximation of the Music Signal

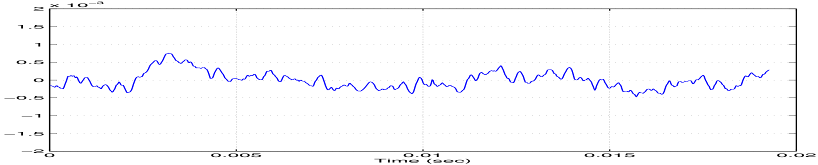
- Randomly pick the $N = 10,000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{\boldsymbol{f}} \in \mathbb{R}^m$.



- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{\boldsymbol{f}}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800,791$ principal components.
- Do the linear and nonlinear approximations as before.

PCA-based Approximation of the Music Signal

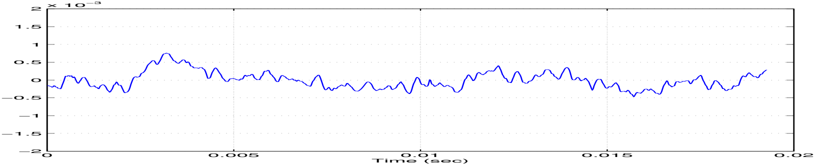
- Randomly pick the $N = 10,000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{\boldsymbol{f}} \in \mathbb{R}^m$.



- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{\boldsymbol{f}}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800,791$ principal components.
- Do the linear and nonlinear approximations as before.

PCA-based Approximation of the Music Signal

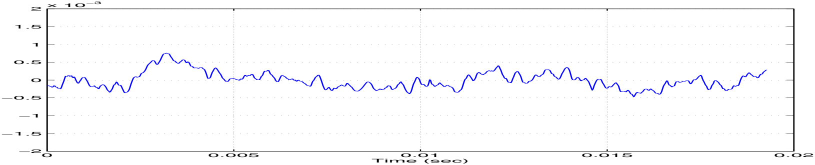
- Randomly pick the $N = 10{,}000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{\boldsymbol{f}} \in \mathbb{R}^m$.



- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{\boldsymbol{f}}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800{,}791$ principal components.
- Do the linear and nonlinear approximations as before.

PCA-based Approximation of the Music Signal

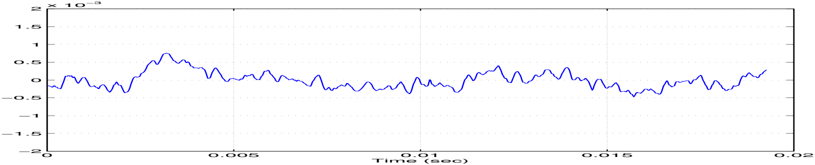
- Randomly pick the $N = 10,000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{\boldsymbol{f}} \in \mathbb{R}^m$.



- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{\boldsymbol{f}}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800,791$ principal components.
- Do the linear and nonlinear approximations as before.

PCA-based Approximation of the Music Signal

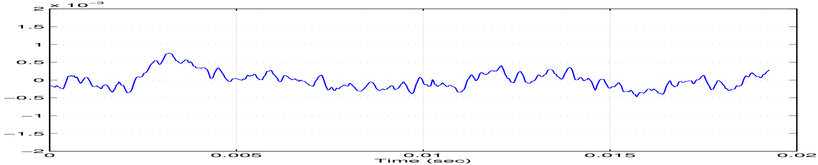
- Randomly pick the $N = 10,000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{\boldsymbol{f}} \in \mathbb{R}^m$.



- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{\boldsymbol{f}}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800,791$ principal components.
- Do the linear and nonlinear approximations as before.

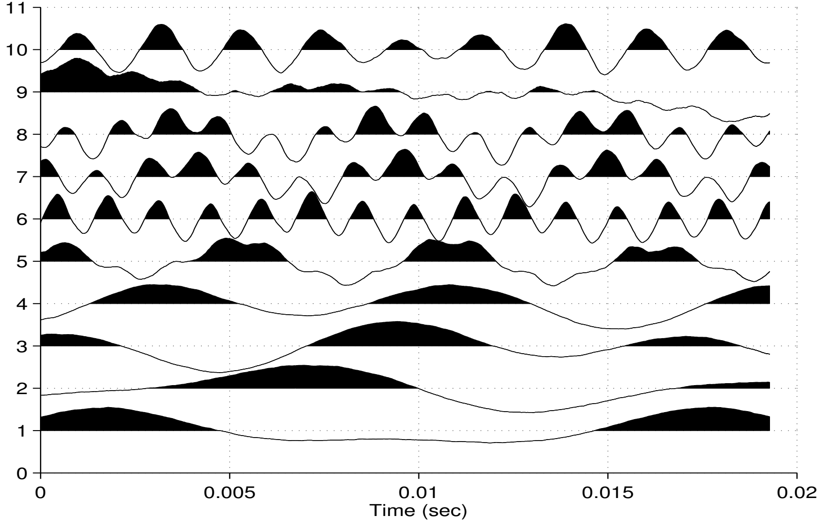
PCA-based Approximation of the Music Signal

- Randomly pick the $N = 10,000$ starting locations of the segments each of which has $m = 851$ samples of the left channel of the Music Signal.
- Compute the mean vector $\overline{\boldsymbol{f}} \in \mathbb{R}^m$.



- Subtract the mean vector from each column of the data matrix $F \in \mathbb{R}^{m \times N}$, i.e., compute the *centered* data matrix $\widetilde{F} := F - \overline{\boldsymbol{f}}\mathbf{1}^{\mathsf{T}}$.
- Compute $W_{\mathrm{PCA}} \in \mathbb{R}^{m \times m}$ of $\widetilde{F}$ via SVD.
- Chop the original signal into $N = 941$ mutually exclusive segments of length $m = 851$, and multiply $W_{\mathrm{PCA}}^{\mathsf{T}}$ with each segment to compute the principal components (i.e., expansion coefficients) of that segment, which gives us $m \times N = 800,791$ principal components.
- Do the linear and nonlinear approximations as before.

The First 10 PCA Basis Vectors



I. Linear Approximation with PCA

In each segment, only retain the first k principal components out of m components, which gives us $k \times N$ principal components in total; Increment or decrement k at some segments to have the n' total retained principal components; reconstruct all the segments from the retained components.

I. Linear Approximation with PCA

In each segment, only retain the first k principal components out of m components, which gives us $k \times N$ principal components in total; Increment or decrement k at some segments to have the n' total retained principal components; reconstruct all the segments from the retained components.

I. Linear Approximation with PCA

In each segment, only retain the first k principal components out of m components, which gives us $k \times N$ principal components in total; Increment or decrement k at some segments to have the n' total retained principal components; reconstruct all the segments from the retained components.

I. Linear Approximation with PCA

In each segment, only retain the first k principal components out of m components, which gives us $k \times N$ principal components in total; Increment or decrement k at some segments to have the n' total retained principal components; reconstruct all the segments from the retained components.

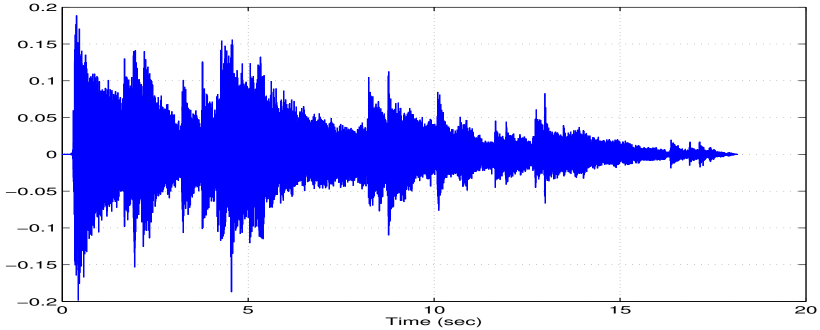


Figure: The first 100,000 principal components are retained (only 12.5% of the whole coefficients): Aja: the first 100,000 PC's

I. Linear Approximation with PCA

In each segment, only retain the first k principal components out of m components, which gives us $k \times N$ principal components in total; Increment or decrement k at some segments to have the n' total retained principal components; reconstruct all the segments from the retained components.

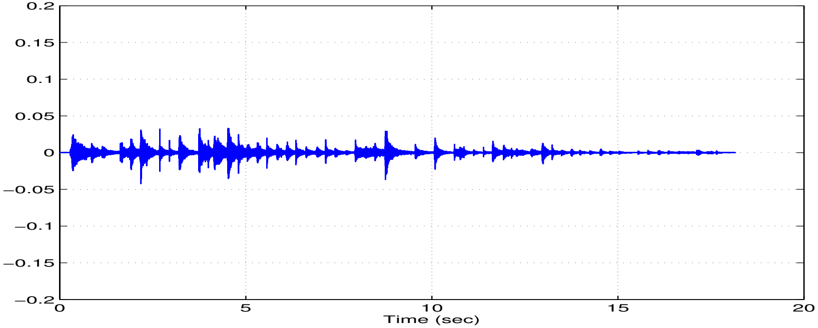


Figure: Diff. from the original: Aja: the remaining 700,791 PC's

I. Linear Approximation with PCA

In each segment, only retain the first k principal components out of m components, which gives us $k \times N$ principal components in total; Increment or decrement k at some segments to have the n' total retained principal components; reconstruct all the segments from the retained components.

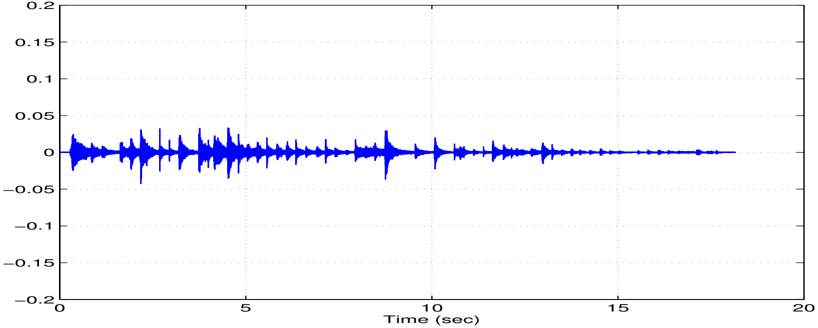


Figure: Diff. from the original: Aja: the remaining 700,791 PC's

This approximation with PCA is better than that with the global DCT!

II. Nonlinear Approximation with PCA

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

II. Nonlinear Approximation with PCA

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

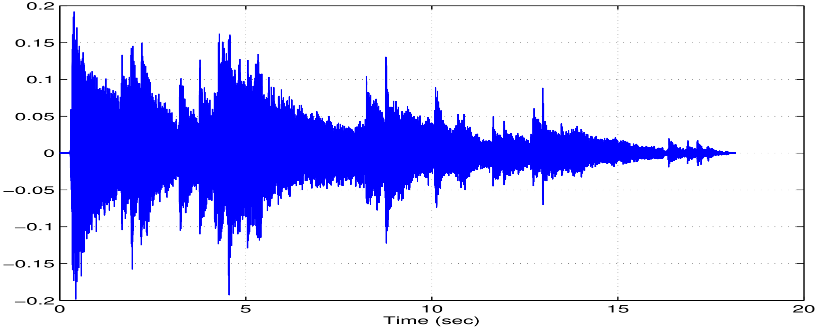


Figure: The largest 100,000 principal components are retained (only 12.5% of the whole coefficients): Aja: the largest 100,000 PC's

II. Nonlinear Approximation with PCA

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

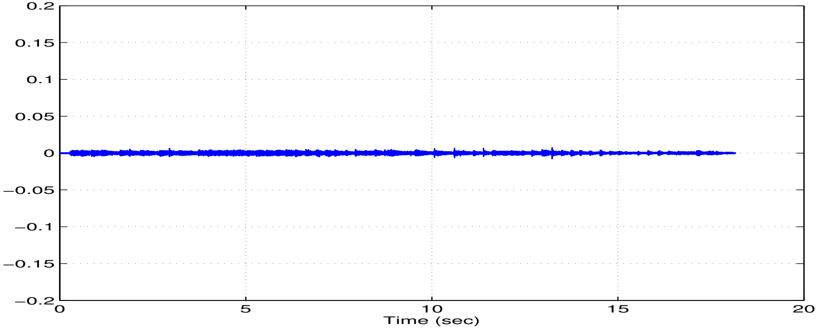


Figure: Diff. from the original: Aja: the smallest 700,791 PC's

II. Nonlinear Approximation with PCA

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

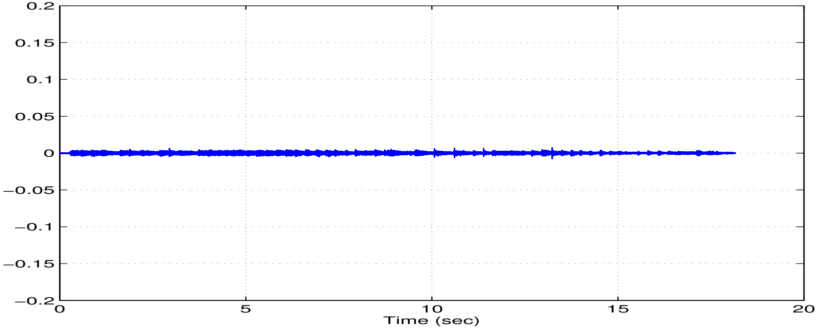


Figure: Diff. from the original: Aja: the smallest 700,791 PC's

This nonlinear approximation is the best so far!

Further Remarks on PCA

- Computing PCA/KLT by chopping a given music signal and using the resulting basis for approximation/compression is not computationally efficient; storing the basis vectors also costs the storage space.
- If the underlying stochastic process obeys the (high-dimensional) *Gaussian (or normal) distribution*, then PCA/KLT provides not only the decorrelated coordinates but also the *statistically-independent* coordinates!
- Again, the quality of the PCA/KLT performance depends on the number of available realizations N and the dimension of the process n. The classical setting of $n \ll N$ is favorable for PCA/KLT.
- The *mean-squared error minimization* is a mathematical convenience; it may not be necessarily the best criterion in terms of perceptual quality assessed by humans.

Further Remarks on PCA

- Computing PCA/KLT by chopping a given music signal and using the resulting basis for approximation/compression is not computationally efficient; storing the basis vectors also costs the storage space.
- If the underlying stochastic process obeys the (high-dimensional) *Gaussian (or normal) distribution*, then PCA/KLT provides not only the decorrelated coordinates but also the *statistically-independent* coordinates!
- Again, the quality of the PCA/KLT performance depends on the number of available realizations N and the dimension of the process n. The classical setting of $n \ll N$ is favorable for PCA/KLT.
- The *mean-squared error minimization* is a mathematical convenience; it may not be necessarily the best criterion in terms of perceptual quality assessed by humans.

Further Remarks on PCA

- Computing PCA/KLT by chopping a given music signal and using the resulting basis for approximation/compression is not computationally efficient; storing the basis vectors also costs the storage space.
- If the underlying stochastic process obeys the (high-dimensional) *Gaussian (or normal) distribution*, then PCA/KLT provides not only the decorrelated coordinates but also the *statistically-independent* coordinates!
- Again, the quality of the PCA/KLT performance depends on the number of available realizations N and the dimension of the process n. The classical setting of $n \ll N$ is favorable for PCA/KLT.
- The *mean-squared error minimization* is a mathematical convenience; it may not be necessarily the best criterion in terms of perceptual quality assessed by humans.

Further Remarks on PCA

- Computing PCA/KLT by chopping a given music signal and using the resulting basis for approximation/compression is not computationally efficient; storing the basis vectors also costs the storage space.
- If the underlying stochastic process obeys the (high-dimensional) *Gaussian (or normal) distribution*, then PCA/KLT provides not only the decorrelated coordinates but also the *statistically-independent* coordinates!
- Again, the quality of the PCA/KLT performance depends on the number of available realizations N and the dimension of the process n. The classical setting of $n \ll N$ is favorable for PCA/KLT.
- The *mean-squared error minimization* is a mathematical convenience; it may not be necessarily the best criterion in terms of perceptual quality assessed by humans.

Outline

Block Discrete Cosine Transform (BDCT)

- Applying DCT locally on the chopped segments is much more efficient in computation; and the basis vectors do not need to be stored (can be computed on the fly thanks to the FFT-based DCT algorithm).
- This transform is called the Block Discrete Cosine Transform (BDCT).
- The 2D version of the BDCT is the de facto standard adopted in the popular *JPEG Image Compression Standard*.
- Using BDCT makes more sense than using the global DCT since many signals and images change their *characteristics* locally; this property is often called non-stationarity $\Longrightarrow$ Discrete Wavelet Transforms also allow such localized analysis.

Block Discrete Cosine Transform (BDCT)

- Applying DCT locally on the chopped segments is much more efficient in computation; and the basis vectors do not need to be stored (can be computed on the fly thanks to the FFT-based DCT algorithm).
- This transform is called the Block Discrete Cosine Transform (BDCT).
- The 2D version of the BDCT is the de facto standard adopted in the popular *JPEG Image Compression Standard*.
- Using BDCT makes more sense than using the global DCT since many signals and images change their *characteristics* locally; this property is often called non-stationarity $\Longrightarrow$ Discrete Wavelet Transforms also allow such localized analysis.

Block Discrete Cosine Transform (BDCT)

- Applying DCT locally on the chopped segments is much more efficient in computation; and the basis vectors do not need to be stored (can be computed on the fly thanks to the FFT-based DCT algorithm).
- This transform is called the Block Discrete Cosine Transform (BDCT).
- The 2D version of the BDCT is the de facto standard adopted in the popular *JPEG Image Compression Standard*.
- Using BDCT makes more sense than using the global DCT since many signals and images change their *characteristics* locally; this property is often called non-stationarity $\Longrightarrow$ Discrete Wavelet Transforms also allow such localized analysis.

Block Discrete Cosine Transform (BDCT)

- Applying DCT locally on the chopped segments is much more efficient in computation; and the basis vectors do not need to be stored (can be computed on the fly thanks to the FFT-based DCT algorithm).
- This transform is called the Block Discrete Cosine Transform (BDCT).
- The 2D version of the BDCT is the de facto standard adopted in the popular *JPEG Image Compression Standard*.
- Using BDCT makes more sense than using the global DCT since many signals and images change their *characteristics* locally; this property is often called non-stationarity $\Longrightarrow$ Discrete Wavelet Transforms also allow such localized analysis.

Block Discrete Cosine Transform (BDCT)

- Applying DCT locally on the chopped segments is much more efficient in computation; and the basis vectors do not need to be stored (can be computed on the fly thanks to the FFT-based DCT algorithm).
- This transform is called the Block Discrete Cosine Transform (BDCT).
- The 2D version of the BDCT is the de facto standard adopted in the popular *JPEG Image Compression Standard*.
- Using BDCT makes more sense than using the global DCT since many signals and images change their *characteristics* locally; this property is often called non-stationarity $\Longrightarrow$ Discrete Wavelet Transforms also allow such localized analysis.

Block Discrete Cosine Transform (BDCT)

- Applying DCT locally on the chopped segments is much more efficient in computation; and the basis vectors do not need to be stored (can be computed on the fly thanks to the FFT-based DCT algorithm).
- This transform is called the Block Discrete Cosine Transform (BDCT).
- The 2D version of the BDCT is the de facto standard adopted in the popular *JPEG Image Compression Standard*.
- Using BDCT makes more sense than using the global DCT since many signals and images change their *characteristics* locally; this property is often called non-stationarity $\Longrightarrow$ Discrete Wavelet Transforms also allow such localized analysis.

Method	Linear	Nonlinear
STD	0.7923	0.4955
DCT (global)	0.1292	0.0941
PCA (local)	0.0985	0.0357
BDCT (local)	0.1291	0.0267

Table: Approximation errors measured in the relative ℓ^2 error

Block Discrete Cosine Transform (BDCT)

- Applying DCT locally on the chopped segments is much more efficient in computation; and the basis vectors do not need to be stored (can be computed on the fly thanks to the FFT-based DCT algorithm).
- This transform is called the Block Discrete Cosine Transform (BDCT).
- The 2D version of the BDCT is the de facto standard adopted in the popular *JPEG Image Compression Standard*.
- Using BDCT makes more sense than using the global DCT since many signals and images change their *characteristics* locally; this property is often called non-stationarity $\Longrightarrow$ Discrete Wavelet Transforms also allow such localized analysis.

Method	Linear	Nonlinear
STD	2.022	6.099
DCT (global)	17.77	20.53
PCA (local)	20.13	28.95
BDCT (local)	17.78	31.47

Table: Approximation errors measured in the Signal-to-Noise Ratio (dB)

I. Linear Approximation with BDCT

In each segment, only retain the first k BDCT coefficients out of m coefficient, which gives us $k \times N$ BDCT coefficients in total; Increment or decrement k at some segments to have the n' total retained BDCT coefficients; reconstruct all the segments from the retained coefficients.

I. Linear Approximation with BDCT

In each segment, only retain the first k BDCT coefficients out of m coefficient, which gives us $k \times N$ BDCT coefficients in total; Increment or decrement k at some segments to have the n' total retained BDCT coefficients; reconstruct all the segments from the retained coefficients.

I. Linear Approximation with BDCT

In each segment, only retain the first k BDCT coefficients out of m coefficient, which gives us $k \times N$ BDCT coefficients in total; Increment or decrement k at some segments to have the n' total retained BDCT coefficients; reconstruct all the segments from the retained coefficients.

I. Linear Approximation with BDCT

In each segment, only retain the first k BDCT coefficients out of m coefficient, which gives us $k \times N$ BDCT coefficients in total; Increment or decrement k at some segments to have the n' total retained BDCT coefficients; reconstruct all the segments from the retained coefficients.

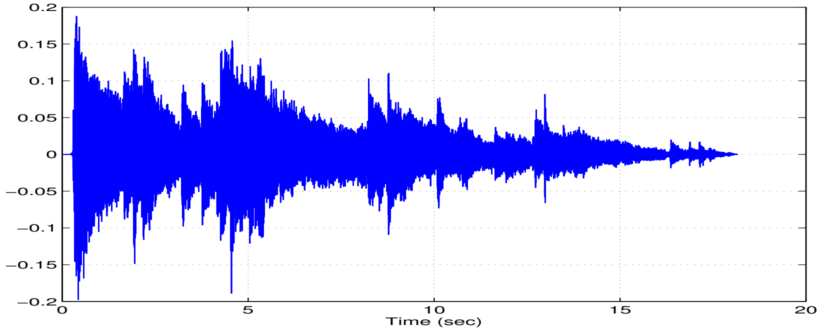


Figure: The first 100,000 principal components are retained (only 12.5% of the whole coefficients): Aja: the first 100,000 BDCT coeff's

I. Linear Approximation with BDCT

In each segment, only retain the first k BDCT coefficients out of m coefficient, which gives us $k \times N$ BDCT coefficients in total; Increment or decrement k at some segments to have the n' total retained BDCT coefficients; reconstruct all the segments from the retained coefficients.

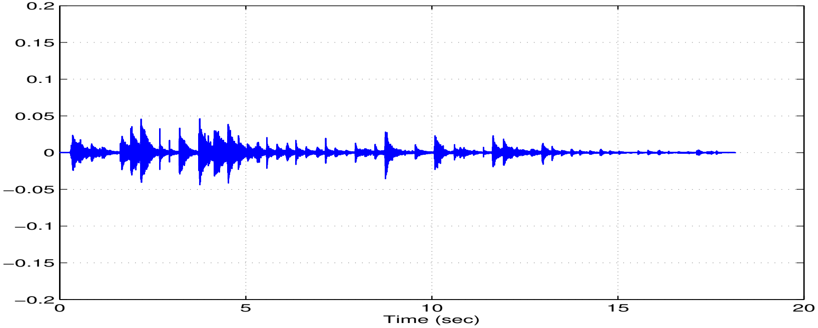


Figure: Diff. from the original: Aja: the remaining 700,791 BDCT coeff's

I. Linear Approximation with BDCT

In each segment, only retain the first k BDCT coefficients out of m coefficient, which gives us $k \times N$ BDCT coefficients in total; Increment or decrement k at some segments to have the n' total retained BDCT coefficients; reconstruct all the segments from the retained coefficients.

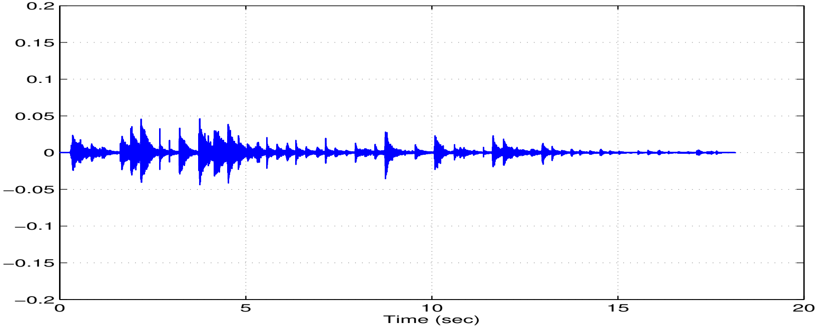


Figure: Diff. from the original: Aja: the remaining 700,791 BDCT coeff's

This approximation with BDCT is better than that with the global DCT!

II. Nonlinear Approximation with BDCT

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

II. Nonlinear Approximation with BDCT

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

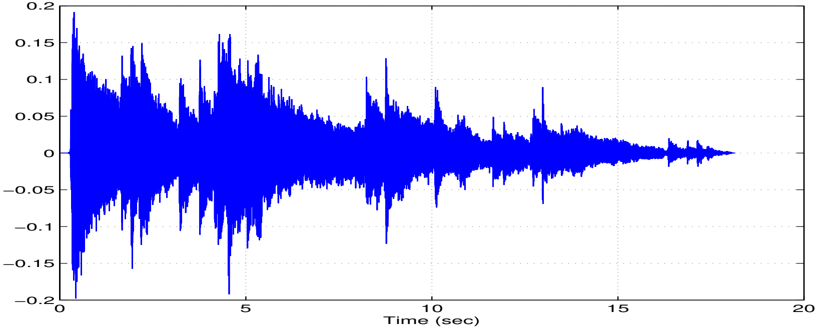


Figure: The largest 100,000 principal components are retained (only 12.5% of the whole coefficients): Aja: the largest 100,000 BDCT coeff's

II. Nonlinear Approximation with BDCT

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

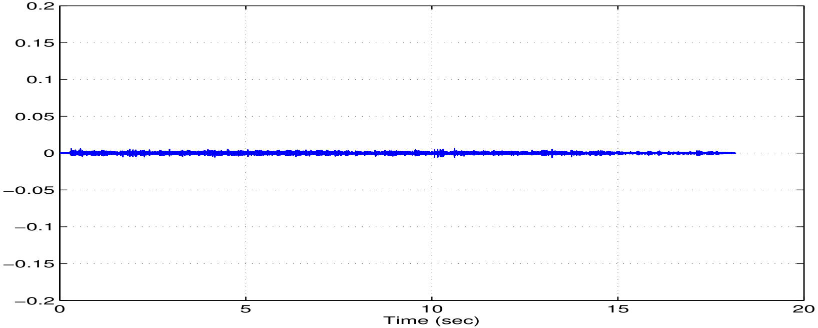


Figure: Diff. from the original: Aja: the smallest 700,791 BDCT coeff's

II. Nonlinear Approximation with BDCT

Only retain the n' largest principal components (in absolute value) of all the segments; reconstruct all the segments from the retained components.

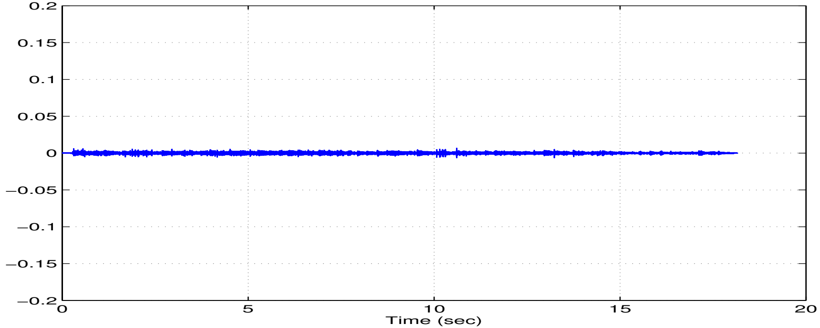


Figure: Diff. from the original: Aja: the smallest 700,791 BDCT coeff's

This nonlinear approximation is the best among all the experiments I conducted for this lecture!

Outline

Reality of Audio Compression

The real compression method used in the mp3 standard is quite involved:

- Processing by the so-called *Modified DCT* (DCT-Type IV with half-overlap and a smoothing window) to reduce the *edge effects*.
- *Quantization* of the expansion coefficients using psychoacoustic models
- *Efficient encoding* of the resulting quantized coefficients (e.g., the Huffman coding; the arithmetic coding, . . .)

Reality of Audio Compression

The real compression method used in the mp3 standard is quite involved:

- Processing by the so-called *Modified DCT* (DCT-Type IV with half-overlap and a smoothing window) to reduce the *edge effects*.
- *Quantization* of the expansion coefficients using psychoacoustic models
- *Efficient encoding* of the resulting quantized coefficients (e.g., the Huffman coding; the arithmetic coding, . . .)

Reality of Audio Compression

The real compression method used in the mp3 standard is quite involved:

- Processing by the so-called *Modified DCT* (DCT-Type IV with half-overlap and a smoothing window) to reduce the *edge effects*.
- *Quantization* of the expansion coefficients using psychoacoustic models
- *Efficient encoding* of the resulting quantized coefficients (e.g., the Huffman coding; the arithmetic coding, . . .)

Reality of Audio Compression

The real compression method used in the mp3 standard is quite involved:

- Processing by the so-called *Modified DCT* (DCT-Type IV with half-overlap and a smoothing window) to reduce the *edge effects*.
- *Quantization* of the expansion coefficients using psychoacoustic models
- *Efficient encoding* of the resulting quantized coefficients (e.g., the Huffman coding; the arithmetic coding, . . .)

Outline

The Roadmap to Graphs & Networks

- Usual digital signals and images are recorded on the so-called regular lattice.
- All these great *harmonic analysis tools* (DCT, Wavelets, ...) have been developed on data recorded on such regular lattices.
- Now, due to the advent of sensor technology and social network infrastructure, more and more data are recorded on quite *irregular* and *non-lattice* sample points, which can be represented by graphs and networks.
- A big question: How can we transfer those harmonic analysis tools for data recorded on graphs and networks?

The Roadmap to Graphs & Networks

- Usual digital signals and images are recorded on the so-called regular lattice.
- All these great *harmonic analysis tools* (DCT, Wavelets, . . .) have been developed on data recorded on such regular lattices.
- Now, due to the advent of sensor technology and social network infrastructure, more and more data are recorded on quite *irregular* and *non-lattice* sample points, which can be represented by graphs and networks.
- A big question: How can we transfer those harmonic analysis tools for data recorded on graphs and networks?

The Roadmap to Graphs & Networks

- Usual digital signals and images are recorded on the so-called regular lattice.
- All these great *harmonic analysis tools* (DCT, Wavelets, . . .) have been developed on data recorded on such regular lattices.
- Now, due to the advent of sensor technology and social network infrastructure, more and more data are recorded on quite *irregular* and *non-lattice* sample points, which can be represented by graphs and networks.
- A big question: How can we transfer those harmonic analysis tools for data recorded on graphs and networks?

The Roadmap to Graphs & Networks

- Usual digital signals and images are recorded on the so-called regular lattice.
- All these great *harmonic analysis tools* (DCT, Wavelets, . . .) have been developed on data recorded on such regular lattices.
- Now, due to the advent of sensor technology and social network infrastructure, more and more data are recorded on quite *irregular* and *non-lattice* sample points, which can be represented by graphs and networks.
- A big question: How can we transfer those harmonic analysis tools for data recorded on graphs and networks?