Outline

1. Student Learning Objectives
   - SLOs: Orthogonal Projections and Orthonormal Bases
   - Orthogonal Projections and Orthonormal Bases
   - Suggested Problems 5.1
   - Lecture – Book Roadmap
   - Supplemental Material
   - Why Orthonormality Matters
   - Application: The (Fast) Fourier Transform
   - Application: MPEG-4 Compression without some of the Math

2. Orthogonality and Least Squares
   - Suggested Problems
   - Suggested Problems 5.1
   - Lecture – Book Roadmap

3. Suggested Problems
   - Suggested Problems 5.1
   - Lecture – Book Roadmap

4. Suplemental Material
   - Metacognitive Reflection
   - Problem Statements 5.1

5. Why Orthonormality Matters
   - Application: The (Fast) Fourier Transform
   - Application: MPEG-4 Compression without some of the Math

---

Student Learning Objectives SLOs: Orthogonal Projections and Orthonormal Bases

After this lecture you should:

- Understand the concept of Orthonormality.
- Be able to compute the projection onto a subspace \( V \) (with \( \dim(V) = 1 \)), using an orthonormal basis.
- Be comfortable with the use of the Orthogonal Complement, \( V^\perp \) of a subspace \( V \).

Orthogonal Projections and Orthonormal Bases

Definition (Orthogonality, Length, Unit Vectors)

- Two vectors \( \vec{v} \) and \( \vec{w} \in \mathbb{R}^n \) are called orthogonal (or perpendicular) if \( \vec{v} \cdot \vec{w} = 0 \).
- The length (or norm, or magnitude) of a vector \( \vec{v} \in \mathbb{R}^n \) is \( \|\vec{v}\| = \sqrt{\vec{v} \cdot \vec{v}} \).
- A vector \( \vec{u} \in \mathbb{R}^n \) is called a unit vector if \( \|\vec{u}\| = 1 \).

Definition (Orthonormal Vectors)

The vectors \( \vec{u}_1, \ldots, \vec{u}_m \in \mathbb{R}^n \) are called orthonormal if they are all unit vectors and orthogonal to one another:

\[
\vec{u}_i \cdot \vec{u}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}
\]
Orthogonal Projections and Orthonormal Bases

Orthonormal Vectors

Example (The “Standard Basis Vectors”)
The vectors $\vec{e}_1, \ldots, \vec{e}_n \in \mathbb{R}^n$ are orthonormal. — They form an orthonormal basis for $\mathbb{R}^n$.

Example (Rotated Standard Vectors in $\mathbb{R}^2$)
Consider $\vec{e}_1$ and $\vec{e}_2$ in $\mathbb{R}^2$; and their rotated versions:

$$\vec{r}_1(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix},$$

$$\vec{r}_2(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix},$$

in $\mathbb{R}^4$ are orthonormal.

Unit Length: $\sqrt{4 \times \frac{1}{2^2}} = 1$.

Orthogonality: For each pair of vectors, two of the products $\vec{u}_i(k)\vec{u}_j(k)$ will be positive and two negative; hence the sum $\sum_{k=1}^4 \vec{u}_i(k)\vec{u}_j(k)$ will be zero.

Properties

Theorem (Properties of Orthonormal Vectors)

a. Orthonormal Vectors are linearly independent.
b. Orthonormal Vectors $\vec{u}_1, \ldots, \vec{u}_n \in \mathbb{R}^n$ form a basis of $\mathbb{R}^n$.

a. Clearly, there is no way to (linearly) combine perpendicular vectors to describe each other (for example, think of $\vec{e}_1$, $\vec{e}_2$, and $\vec{e}_3$ in $\mathbb{R}^3$.)

b. By previous theorems, $n$ linearly independent vectors in $\mathbb{R}^n$ necessarily form a basis of $\mathbb{R}^n$. (Think of the standard basis $\vec{e}_1, \ldots, \vec{e}_n \in \mathbb{R}^n$; and rotations / reflections of it...)

Theorem (Orthogonal Projection)

Consider a vector $\vec{x} \in \mathbb{R}^n$ and a subspace $V$ of $\mathbb{R}^n$. Then we can write

$$\vec{x} = \vec{x}^\parallel + \vec{x}^\perp,$$

where $\vec{x}^\parallel \in V$, and $\vec{x}^\perp \perp V$. This representation is unique. The vector $\vec{x}^\parallel$ is called the orthogonal projection of $\vec{x}$ onto $V$, sometimes denoted $\text{proj}_V(\vec{x})$; the transformation $T(\vec{x}) = \text{proj}_V(\vec{x}) = \vec{x}^\parallel$ from $\mathbb{R}^n \to \mathbb{R}^n$ is linear.

We have discussed this previously, but only in the context of describing the image and kernel of the projection... We are now ready to start discussing HOW we can compute the projection in any dimension space/subspace.
Orthogonal Projections and Orthonormal Bases

Orthogonal Projection: Formula

Theorem (Formula for the Orthogonal Projection)

If $V$ is a subspace of $\mathbb{R}^n$ with an orthonormal basis $\vec{u}_1, \ldots, \vec{u}_m$, then

$$\text{proj}_V(\vec{x}) = \vec{x} \parallel = (\vec{u}_1 \cdot \vec{x})\vec{u}_1 + \cdots + (\vec{u}_m \cdot \vec{x})\vec{u}_m$$

$\forall \vec{x} \in \mathbb{R}^n$.

Having the orthonormal basis is the absolute key to this formula. Any other basis will produce strange (incorrect) results.

Orthogonal Projection: Example

Example (Orthogonal Projection (part 1))

Consider the subspace $V = \text{im}(A)$ of $\mathbb{R}^4$, find $\text{proj}_V(\vec{x})$; where

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} \pm 1 \\ \pm 1 \\ \pm 1 \end{bmatrix} = \sqrt{4(\pm 1)^2} = \sqrt{4} = 2.$$

Since the columns $\vec{a}_1$ and $\vec{a}_2$ are linearly independent, and orthogonal (zero dot-product), they form an orthogonal basis of $V$. Dividing each vector by its length gives us an orthonormal basis for $V = \text{span}(\vec{u}_1, \vec{u}_2)$, where

$$\vec{u}_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}, \quad \vec{u}_2 = \begin{bmatrix} -1/2 \\ -1/2 \\ -1/2 \\ -1/2 \end{bmatrix}.$$

Continued on the next slide...

Example (Orthogonal Projection (part 2))

Now we use the projection formula,

$$\begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 1 \\ 7 \end{bmatrix} = \boxed{6}, \quad \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \boxed{2},$$

therefore

$$\text{proj}_V(\vec{x}) = 6\vec{u}_1 + 2\vec{u}_2 = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 2 \\ 4 \end{bmatrix}.$$
**Orthogonal Projection onto a Basis**

**Theorem (Orthogonal Projection onto a Basis)**

Consider an orthonormal basis \( \vec{u}_1, \ldots, \vec{u}_n \) of \( \mathbb{R}^n \). Then

\[
\vec{x} = (\vec{u}_1 \cdot \vec{x})\vec{u}_1 + \cdots + (\vec{u}_n \cdot \vec{x})\vec{u}_n
\]

\( \forall \vec{x} \in \mathbb{R}^n \).

Since the basis spans \( \mathbb{R}^n \), we can “rebuild” \( \vec{x} \) completely by adding up all the projected pieces.

We have \( \vec{x} \) as a **unique linear combination**

\[
\vec{x} = c_1 \vec{u}_1 + \cdots + c_n \vec{u}_n
\]

where \( c_k = \vec{u}_k \cdot \vec{x} \).

**Image, and Kernel...**

Using our recently acquired vocabulary, we realize that

\[
\text{im}(\text{proj}_V(\vec{x})) = V
\]

Also, we know we can write

\[
\vec{x} = \vec{x}^\parallel + \vec{x}^\perp
\]

and \( \text{proj}_V(\vec{x}) = \vec{x}^\parallel \), so if we are looking for the kernel, we want all \( \vec{x} \) without a \( \vec{x}^\parallel \) part, *i.e.* \( \{ \vec{x} \in \mathbb{R}^n : \vec{x} = \vec{x}^\perp \} \), the collection of all vectors orthogonal to the subspace \( V \).

A formal definition follows on the next slide...

**The Orthogonal Complement :: Definition and Related Expressions**

**Definition (Orthogonal Complement)**

Consider a subspace \( V \) of \( \mathbb{R}^m \). The **Orthogonal Complement** \( V^\perp \) of \( V \) is the set of those vectors \( \vec{x} \in \mathbb{R}^m \) that are orthogonal to all vectors in \( V \):

\[
V^\perp = \{ \vec{x} \in \mathbb{R}^m : \vec{x} \cdot \vec{v} = 0, \forall \vec{v} \in V \}
\]

Note* that \( V^\perp \) is the kernel of \( \text{proj}_V(\vec{x}) \).

* This means that if we have a description of \( V \) as the solution of a linear system \( (A \in \mathbb{R}^{n \times m}) \)

\[
V = \{ \vec{x} \in \mathbb{R}^m : A\vec{x} = \vec{0} \}
\]

then (note that \( \text{proj}_V(\vec{x}) : \mathbb{R}^m \to \mathbb{R}^m \))

\[
V = \ker(A) = \text{im}(\text{proj}_V(\vec{x})) \subset \mathbb{R}^m
\]

\[
V^\perp = \ker(A^\top) = \text{im}(\text{proj}_V(\vec{x})) \subset \mathbb{R}^m
\]
Consider a general linear transformation $T(\vec{x}) = A\vec{x}$, where $A \in \mathbb{R}^{n \times m}$.

<table>
<thead>
<tr>
<th>“Input Space”</th>
<th>“Output Space”</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{x} \in \mathbb{R}^n$</td>
<td>$\vec{y} = A\vec{x} \in \mathbb{R}^n$</td>
</tr>
<tr>
<td>$\ker(A)$</td>
<td>$\vec{0}$</td>
</tr>
<tr>
<td>$\ker(A)^\perp$</td>
<td>$\text{im}(A)$</td>
</tr>
<tr>
<td>nothing</td>
<td>$\text{im}(A)^\perp$</td>
</tr>
</tbody>
</table>

$\ker(A) \oplus \ker(A)^\perp = \mathbb{R}^n$ \hspace{1cm} $\text{im}(A) \oplus \text{im}(A)^\perp = \mathbb{R}^n$

**Theorem (Properties of the Orthogonal Complement)**

- **a.** The Orthogonal Complement $V^\perp$ of $V$ is a subspace of $\mathbb{R}^n$.
- **b.** The intersection (common elements) of $V^\perp$ and $V$ consists of the zero vector: $V^\perp \cap V = \{ \vec{0} \}$. \(\forall \vec{x} \in V^\perp \cap V : \vec{x} \cdot \vec{x} = 0 \Rightarrow \vec{x} = \vec{0} \).\n- **c.** $\dim(V) + \dim(V^\perp) = n$. [By Rank-Nullity Theorem: $T(\vec{x}) = \text{proj}_V(\vec{x})$]
- **d.** $(V^\perp)^\perp = V$.
- **e.** The “direct sum” $V \oplus V^\perp = \mathbb{R}^n$, where

$$U = V \oplus W \overset{\text{def}}{=} \{ \vec{u} = \vec{v} + \vec{w} : \vec{v} \in V, \vec{w} \in W \},$$

that is, $V$ and $V^\perp$ “split” the space in two non-overlapping parts — in this context $\vec{0}$ does not “count” as an overlap.

**From Pythagoras to Cauchy-(Bunyakovsky)-Schwarz**

**Example (Really Old Stuff in New Notation)**

Consider a line $L$, and a vector $\vec{x}$ in $\mathbb{R}^n$. If we project $\vec{x}$ onto $L$ and write $\vec{x} = \vec{x}^\parallel + \vec{x}^\perp$ good ole’ Pythagoras says

$$\|\vec{x}\|^2 \equiv \|\vec{x}^\parallel + \vec{x}^\perp\|^2 = \|\vec{x}^\parallel\|^2 + \|\vec{x}^\perp\|^2.$$ 

Checking:

$$\|\vec{x}\|^2 = \|\vec{x}^\parallel + \vec{x}^\perp\|^2$$
$$= (\vec{x}^\parallel + \vec{x}^\perp) \cdot (\vec{x}^\parallel + \vec{x}^\perp)$$
$$= \vec{x}^\parallel \cdot \vec{x}^\parallel + \vec{x}^\parallel \cdot \vec{x}^\perp + \vec{x}^\perp \cdot \vec{x}^\parallel + \vec{x}^\perp \cdot \vec{x}^\perp$$
$$= \|\vec{x}^\parallel\|^2 + 0 + 0 + \|\vec{x}^\perp\|^2$$

... and there it is!

A. **Pythagorean Theorem**

Consider two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$. The equation

$$\|\vec{x} + \vec{y}\|^2 = \|\vec{x}\|^2 + \|\vec{y}\|^2$$

holds if and only if $\vec{x} \perp \vec{y}$.

B. **Theorem (proj$_V(\vec{x})$ is no longer than $\vec{x}$)**

Consider a subspace $V$ of $\mathbb{R}^n$, and a vector $\vec{x} \in \mathbb{R}^n$. Then

$$\|\text{proj}_V(\vec{x})\| \leq \|\vec{x}\|$$

where equality is achieved if and only if $\vec{x} \in V$.  

Peter Blomgren, (blomgren.peter@gmail.com)
From Pythagoras to Cauchy-(Bunyakovsky)-Schwarz

Proof: (by observation).

Since

\[ \| \vec{x}^\perp \|^2 + \| \vec{x}^\parallel \|^2 = \| \vec{x} \|^2 \]

(and all lengths are non-negative), we must have

\[ \| \vec{x}^\perp \|^2 \leq \| \vec{x} \|^2. \]

The Dot Product, \( \cos(\theta) \), and Cauchy-Schwarz

Consider two vectors \( \vec{x}, \vec{y} \in \mathbb{R}^n \). We have previously expressed the dot product in terms of the angle between the two vectors.

\[ \vec{x} \cdot \vec{y} = \cos(\theta) \| \vec{x} \| \| \vec{y} \| \]

The real use of the formula is to find

\[ \cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \| \| \vec{y} \|}. \quad \theta = \arccos\left( \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \| \| \vec{y} \|} \right) \]

Cauchy-Bunyakovsky-Schwarz, \( |\vec{x} \cdot \vec{y}| \leq \| \vec{x} \| \| \vec{y} \| \), guarantees that the argument to \( \arccos \), and the value of \( \cos(\theta) \) (as defined here) make sense.

Note that the angle \( \theta \) is in the plane spanned by \( \vec{x} \), and \( \vec{y} \).
Example: Angle Between Vectors

Find the angle between
\[ \vec{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \]

We have
\[ \|\vec{x}\| = 1, \quad \|\vec{y}\| = \sqrt{4} = 2, \quad \vec{x} \cdot \vec{y} = 1 \]

so that
\[ \cos(\theta) = \frac{1}{2}, \quad \theta = \arccos \left( \frac{1}{2} \right) = \frac{\pi}{3} \]
(5.1.7) For vectors \( \vec{u}, \vec{v} \), determine whether the angle is acute \((< \frac{\pi}{2})\), right \((= \frac{\pi}{2})\), or obtuse \((> \frac{\pi}{2})\).

\[ \vec{u} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}. \]

(5.1.10) For which value(s) of \( k \in \mathbb{R} \) are the vectors

\[ \vec{u} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} 1 \\ k \\ 1 \end{bmatrix} \]

perpendicular?

(5.1.11) Consider the vectors

\[ \vec{u} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \vec{v} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^n \]

a. For \( n = 2, 3, 4 \), find the angle \( \theta_n \) between \( \vec{u} \) and \( \vec{v} \).

b. Find the limit of \( \theta_n \) as \( n \to \infty \).

(5.1.15) Consider the vector

\[ \vec{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \in \mathbb{R}^4. \]

Find a basis for the subspace of \( \mathbb{R}^4 \) consisting of all vectors perpendicular to \( \vec{v} \).

(5.1.17) Find a basis for \( W^\perp \), where

\[ W = \text{span} \left( \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} \right). \]

(5.1.27) Find the orthogonal projection of \( 9\vec{e}_1 \) onto the subspace \( W \) of \( \mathbb{R}^4 \), where

\[ W = \text{span} \left( \begin{bmatrix} 2 \\ 2 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 2 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right), \quad \vec{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \]
A lot of signal analysis and processing is frequency-based; meaning that it is highly useful to express a signal using basis functions that are determined by various frequencies.

Consider the functions:

\[
\begin{align*}
\Phi_0(x) &= \frac{1}{2} \\
\Phi_k(x) &= \cos(kx), \quad k = 1, \ldots, n \\
\Phi_{n+k}(x) &= \sin(kx), \quad k = 1, \ldots, n-1
\end{align*}
\]

and let each one define a vector \( \vec{v}_i \in \mathbb{R}^{2n} \), by evaluating the function in the points \( x_j = -\pi + (j\pi/n) \), \( j = 0, 1, \ldots, (2m - 1) \).

Since the vectors are orthogonal, solving the system is not necessary; we can get each coefficient by computing a length 2n dot-product:

\[
a_k = \frac{1}{n} \sum_{j=0}^{2n-1} f_j \cos(kx_j), \quad b_k = \frac{1}{n} \sum_{j=0}^{2n-1} f_j \sin(kx_j).
\]

where \( a_k, k = 0, \ldots, n \) are the first \( (n + 1) \) coefficients of \( [\vec{f}]_B \), and \( b_k, k = 1, \ldots, (n-1) \) are the remaining coefficients.

This approach, the Slow Fourier Transform requires roughly \( 4n^2 \) operations.

[FULL DISCLOSURE] We have omitted a few (3) factors of 2 which are necessary to make the vectors orthonormal.
The Super-Slow, Slow, and Fast Fourier Transform

Much of the analysis was done by Jean Baptiste Joseph Fourier in the early 1800s, but the use of the Fourier series representation was not practical until...

1965, when Cooley and Tukey* published a 4-page paper describing an algorithm which computes the coefficients using only $O(n \log_2 n)$ operations.

It is hard to overstate the importance of this paper!!!

The algorithm is now known as the “Fast Fourier Transform” or just the “FFT”. We sweep the details of the FFT under the rug; it comes down to some clever complex analysis, and the facts that $1 + 1 = 2$, and $1 - 1 = 0$.

---


†The “4k Digital Video Format” has a resolution of 4,194,304 × 2,120 pixels; the tentative Ultra High Definition Television (UHDTV) specification calls for 7,680 × 4,320 pixels for 16:9 aspect ratio (1080p, 12 bits/channel (at least 3 – RGB) ~ 4.0 × 10⁹ 36-bit pixels/sec). IMAX shot on 70 mm film has a theoretical pixel resolution of 12,000 × 8,700 (at 24 fps, for a total of 2.5 × 10¹² pixels/sec).

---

Comparing Operation Counts

<table>
<thead>
<tr>
<th>n</th>
<th>SLOW-FT time</th>
<th>FFT time</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>9 s</td>
<td>1 s</td>
<td>9</td>
</tr>
<tr>
<td>64</td>
<td>29 s</td>
<td>1 s</td>
<td>28</td>
</tr>
<tr>
<td>256</td>
<td>1 m 33 s</td>
<td>1 s</td>
<td>93</td>
</tr>
<tr>
<td>1,024</td>
<td>5 m 15 s</td>
<td>1 s</td>
<td>315</td>
</tr>
<tr>
<td>4,096</td>
<td>18 m 12 s</td>
<td>1 s</td>
<td>1,092</td>
</tr>
<tr>
<td>16,384</td>
<td>1:04:26</td>
<td>1 s</td>
<td>3,855</td>
</tr>
<tr>
<td>65,536</td>
<td>3:49:57</td>
<td>1 s</td>
<td>13,797</td>
</tr>
<tr>
<td>262,144</td>
<td>13:52:12</td>
<td>1 s</td>
<td>49,932</td>
</tr>
<tr>
<td>1,048,576</td>
<td>2d+02:39:21</td>
<td>1 s</td>
<td>182,361</td>
</tr>
<tr>
<td>4,194,304</td>
<td>7d+18:24:48</td>
<td>1 s</td>
<td>671,088</td>
</tr>
<tr>
<td>8,388,608</td>
<td>14d+22:29:15</td>
<td>1 s</td>
<td>1,290,555</td>
</tr>
<tr>
<td>16,777,216</td>
<td>28d+18:25:13</td>
<td>1 s</td>
<td>2,485,513</td>
</tr>
<tr>
<td>33,554,432</td>
<td>55d+11:31:30</td>
<td>1 s</td>
<td>4,793,490</td>
</tr>
</tbody>
</table>

FFT Transform: Computational Complexity

---

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩ Orthogonal Projections and Orthonormal Bases — (37/54)
Comparing Operation Counts

Fourier Transform: Timing, assuming $10^{10}$ op/s

<table>
<thead>
<tr>
<th>Operation</th>
<th>Computational Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Slow FT</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>Super Slow FT</td>
<td>$10^{0}$</td>
</tr>
</tbody>
</table>

MPEG-4 Compression:: Main Ideas

We used the idea of video compression to motivate why we should care about orthogonal basis... The discussion was somewhat hand-wavy

This is an attempt at describing the key ideas of video compression, using only concepts from calculus and half a semester of linear algebra. (Good luck to me!)

In order to communicate the main ideas, lots of “minor” details have been swept under the rug; several oversimplifications have been committed, and a few convenient lies have been told.

MPEG-4 Compression:: Description of the Problem — 1080p “HD” Movies

Imagine a 2-hour 1080p24 Movie; where we are showing 24 frames/second, and each frame is $1920 \times 1080$ pixels, each pixel has a bit depth of 8-bits per color (whether that’s Red-Green-Blue, or Y-Cb-Cr, is a discussion for someplace else); but the bottom line is that we have 3 bytes/pixel, so we end of with a raw datastream with

\[
\begin{align*}
3 \quad \text{bytes/pixel} \\
\times \quad 24 \quad \text{frames/second} \\
\times \quad 7200 \quad \text{seconds} \\
\times \quad (1920 \times 1080) \quad \text{pixels/frame} \\
= \quad 1,074,954,240,000 \quad \text{bytes}.
\end{align*}
\]

Now, keeping in mind that a standard dual-layer Blu-ray disc holds a measly 50,050,629,632 bytes of data, we need a compression ratio of 1 : 21.5 in order to fit the movie onto a disk. This means we can only store slightly less than 4.7% of the datastream.

OK, OK, OK, the extended version of Lord of the Rings is 208 minutes; so really we can only fit 2.7% of the datastream...

If you are streaming the movie, even LESS data is getting transmitted.
MPEG-4 Compression :: Movie ⇝ A Single Frame ⇝ Gray-scale

Let’s for a moment restrict our discussion to a single 1920×1080 pixel frame; and for simplicity, let’s make it gray-scale.

Gandalf, looking particularly gray...

MPEG-4 Compression :: Compressing the Single Frame

Next we are going to discuss how we can compress this single snapshot to use only 4.6% storage. This is going to require a little bit of mathematics...

Looking at a Single Horizontal/Vertical Line of the Image:
First, we can consider the image to be constructed out of 1080 lines, each with 1920 pixels; which means we have a collection of 1080 vectors \( \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_{1080} \), each “living it up” in \( \mathbb{R}^{1920} \), we can also (simultaneously) think of the as being constructed out of 1920 columns, each with 1080 (vertical) pixels; giving us vectors \( \vec{c}_1, \vec{c}_2, \ldots, \vec{c}_{1920} \), each “living it up” in \( \mathbb{R}^{1080} \).

What we need are some good (orthonormal) bases for \( \mathbb{R}^{1080} \) and \( \mathbb{R}^{1920} \). It turns out that if we are given an even number, \( 2n \) points, then we can use the 2\( n \) vectors generated by the functions

\[
\begin{align*}
\Phi_0(x) &= \frac{1}{2} \\
\Phi_k(x) &= \cos(kx), \quad k = 1, \ldots, n \\
\Phi_{n+k}(x) &= \sin(kx), \quad k = 1, \ldots, n - 1
\end{align*}
\]

evaluated in the interval \([−\pi, \pi]\), at the equally spaced points \( x_j = -\pi + (j\pi/n), \ j = 0, 1, \ldots, (2n - 1) \). The generated set of vectors are orthonormal!

Now, align the points \( x_j \) with the pixels, numbered from \( j = 0 \) to \( j = (2n-1) \), horizontally or vertically. Let \( p_j \) denote the pixel value (gray-scale intensity). Now, if we let

\[
\begin{align*}
ak &= \frac{1}{n} \sum_{j=0}^{2n-1} f_j \cos(kx_j) \\
b_k &= \frac{1}{n} \sum_{j=0}^{2n-1} f_j \sin(kx_j),
\end{align*}
\]

be the values of the [pixel-vector]–[cos/sin-vector] dot-products. In our language the \( a_k \) and \( b_k \) coefficients are coordinates in the cos/sin-vector basis for \( \mathbb{R}^{2n} \); and given the coordinates, we can fully reconstruct the pixel values:

\[
p_j \equiv S(x_j) = \frac{a_0}{2} + \frac{a_n}{2} \cos(nx_j) + \sum_{k=1}^{n-1} [a_k \cos(kx_j) + b_k \sin(kx_j)].
\]
We now have a set-up where we can go from “image coordinates” to \([\cos/\sin\text{-vector}]\) coordinates (and back) using only dot products. What we have defined is known as the (one dimensional) \textit{Fourier transform}.

\textbf{Back to 2D}

Even though the previous discussion gave us a nice way to build orthonormal bases in one dimension, it is far from clear \textit{why} this is desirable.

Now, consider the image we had of Gandalf; and let's perform the above procedure first in the horizontal direction (which transforms the image into 1080 lines of \([\cos/\sin\text{-vector}]\) coordinates. Next, transform \textit{that} “image” in the vertical direction. This now gives us an “image” of \textit{vertical} \([\cos/\sin\text{-vector}]\) coordinates of (horizontal \([\cos/\sin\text{-vector}]\) of Gandalf). This is known as the two dimensional \textit{Fourier transform}.

Next, we throw away some Fourier Coefficients.

Here, we take the time to figure out what (in magnitude) 4.6% of coefficients are the largest. — We keep those, and discard the rest.

Then we reconstruct the image using only the 4.6%.
MPEG-4 Compression :: Gandalf — Compressed Reconstruction

For more details on how video compression actually is implemented, check out the following

**References:**

- **What is H.264** — [http://www.h264info.com/h264.html](http://www.h264info.com/h264.html)