Orthogonal Projections and Orthonormal Bases — (1/52)
Outline

1 Student Learning Objectives
   • SLOs: Orthogonal Projections and Orthonormal Bases

2 Orthogonality and Least Squares
   • Orthogonal Projections and Orthonormal Bases

3 Why Orthonormality Matters
   • Application: The (Fast) Fourier Transform

4 Suggested Problems
   • Suggested Problems 5.1
   • Lecture–Book Roadmap

5 Supplemental Material
   • Metacognitive Reflection
   • Problem Statements 5.1
   • MPEG-4 Compression without some of the Math
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 $V$. 
Definition (Orthogonality, Length, Unit Vectors) (old)

a. Two vectors $\vec{v}$ and $\vec{w} \in \mathbb{R}^n$ are called orthogonal (or perpendicular) if $\vec{v} \cdot \vec{w} = 0$.

b. The length (or norm, or magnitude) of a vector $\vec{v} \in \mathbb{R}^n$ is $\|\vec{v}\| = \sqrt{\vec{v} \cdot \vec{v}}$.

c. A vector $\vec{u} \in \mathbb{R}^n$ is called a unit vector if $\|\vec{u}\| = 1$.

Definition (Orthonormal Vectors) (new)

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:

$$\vec{u}_i \cdot \vec{u}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$
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:

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

\[
\vec{r}_2(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix},
\]
Orthogonality and Least Squares
Why Orthonormality Matters
Suggested Problems

Orthogonal Projections and Orthonormal Bases

Orthonormal Vectors

Example (Orthonormal Vectors in $\mathbb{R}^4$)

The vectors

$$\vec{u}_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}, \quad \vec{u}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ -1/2 \\ -1/2 \end{bmatrix}, \quad \vec{u}_3 = \begin{bmatrix} 1/2 \\ -1/2 \\ 1/2 \\ -1/2 \end{bmatrix}, \quad \vec{u}_4 = \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix}$$

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) (old)

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 \( \mathbb{R}^n \rightarrow \mathbb{R}^n \) 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.
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

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 7 \end{bmatrix}, \quad \| \begin{bmatrix} \pm 1 \\ \pm 1 \\ \pm 1 \end{bmatrix} \| = \sqrt{4(\pm 1)^2} = \sqrt{4} = 2.$$

Since the columns $\vec{a}_1$ and $\vec{a}_2$ are linearly independent, and orthogonal (zero dot-product), they form a [n 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

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

Continued on the next slide...
Now we use the projection formula,

\[ \vec{x} \cdot \vec{u}_1 = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 7 \end{bmatrix} \cdot \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} = 6, \quad \vec{x} \cdot \vec{u}_2 = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 7 \end{bmatrix} \cdot \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{bmatrix} = 2, \]

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: Example

Example (Orthogonal Projection (part 3))

It is worth noting:

\[ \vec{x} = \begin{bmatrix} 1 \\ 3 \\ 1 \\ 7 \end{bmatrix}, \quad \text{proj}_V(\vec{x}) = \begin{bmatrix} 4 \\ 2 \\ 2 \\ 4 \end{bmatrix}. \]

\[ \|\vec{x}\| = \sqrt{1 + 9 + 1 + 49} = \sqrt{60}, \]

\[ \|\text{proj}_V(\vec{x})\| = \sqrt{16 + 4 + 4 + 16} = \sqrt{40} \]

so that \[ \|\text{proj}_V(\vec{x})\| \leq \|\vec{x}\|. \]

(The projection should NOT be longer than the original vector! We will show this in a few slides...)
Theorem (Orthogonal Projection onto a Basis)

Consider an orthonormal basis $\mathbf{u}_1, \ldots, \mathbf{u}_n$ of $\mathbb{R}^n$. Then

$$\mathbf{\hat{x}} = (\mathbf{u}_1 \cdot \mathbf{x})\mathbf{u}_1 + \cdots + (\mathbf{u}_n \cdot \mathbf{x})\mathbf{u}_n$$

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

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

We have $\mathbf{\hat{x}}$ as a unique linear combination

$$\mathbf{\hat{x}} = c_1 \mathbf{u}_1 + \cdots + c_n \mathbf{u}_n$$

where $c_k = \mathbf{u}_k \cdot \mathbf{\hat{x}}$. 

Orthogonal Projections and Orthonormal Bases — (13/52)
Orthogonal Projection onto a Basis

If we look in the rear-view mirror [3.4: COORDINATES], we can let

\[
\text{Basis: } \mathcal{U} = \langle \vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n \rangle
\]

\[
\text{Coordinates: } [\vec{x}]_{\mathcal{U}} = \begin{bmatrix}
    c_1 \\
    c_2 \\
    \vdots \\
    c_n
\end{bmatrix}
\]

\[c_k = \vec{u}_k \cdot \vec{x}, \quad k = 1, \ldots, n.\]

Orthogonality allows us to compute the coordinates one-at-a-time, i.e., they are independent from each other.
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 (Orthogonal Complement)

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

$$V^\perp = \{ \vec{x} \in \mathbb{R}^n : \vec{x} \cdot \vec{v} = 0, \forall \vec{v} \in V \}$$

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

Theorem (Properties of the Orthogonal Complement)

Consider a subspace $V$ of $\mathbb{R}^n$.

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:

$$V^\perp \cap V = \{ \vec{0} \}. \ [\vec{x} \in V^\perp \cap V : \vec{x} \cdot \vec{x} = 0 \Rightarrow \vec{x} = \vec{0}.]$$

c. $\dim(V) + \dim(V^\perp) = n$. [By Rank-Nullity Theorem; $T(\vec{x}) = \text{proj}_V(\vec{x})$]

d. $(V^\perp)^\perp = V$. 
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

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

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!
Theorem (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 $\vec{x} \perp \vec{y}$.

Theorem ($\text{proj}_V(\vec{x})$ 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 $\vec{x} \in V$. 
Proof.

Since

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

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

\[ \|\vec{x}\|^2 \leq \|\vec{x}\|^2. \]
From Pythagoras to Cauchy-Schwarz

Now, ponder a one-dimensional subspace (a line through the origin) $V$ of $\mathbb{R}^n$, and let $\vec{y}$ be a vector in that subspace; let

$$\vec{u} = \frac{1}{\|\vec{y}\|} \vec{y}$$

be a unit vector spanning $V$. We can now write the projection using $\vec{u}$:

$$\text{proj}_V(\vec{x}) = (\vec{x} \cdot \vec{u}) \vec{u}$$

It follows that

$$\|\vec{x}\| \geq \|\text{proj}_V(\vec{x})\| = \|\vec{x} \cdot \vec{u}\| \|\vec{u}\| = |\vec{x} \cdot \vec{u}| = |\vec{x} \cdot \frac{1}{\|\vec{y}\|} \vec{y}| = \frac{1}{\|\vec{y}\|} |\vec{x} \cdot \vec{y}|$$

so that

$$|\vec{x} \cdot \vec{y}| \leq \|\vec{x}\| \|\vec{y}\|$$

Peter Blomgren, (blomgren.peter@gmail.com)
What we showed on the previous slide is known as:

**Theorem (The Cauchy-(Bunyakovskiy)-Schwarz Inequality)**

If \( \vec{x} \) and \( \vec{y} \) \( \in \mathbb{R}^n \), then

\[
|\vec{x} \cdot \vec{y}| \leq \|\vec{x}\| \|\vec{y}\|.
\]

The statement is an equality *if and only if* \( \vec{x} \) and \( \vec{y} \) are parallel.
Consider two vectors $\vec{x}$, and $\vec{y} \in \mathbb{R}^n$. 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-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

\[ \cos(\theta) = \frac{1}{2}, \quad \theta = \arccos \left( \frac{1}{2} \right) = \frac{\pi}{3} \]
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 \( x_j = -\pi + (j\pi/n), \quad j = 0, 1, \ldots, (2m-1) \).
The Super-Slow, Slow, and Fast Fourier Transform

Let those vectors be the columns in a matrix, $M \in \mathbb{R}^{2n \times 2n}$.

It turns out that the vectors are linearly independent, which makes them a basis, $\mathcal{B}$, for $\mathbb{R}^{2n}$ and the matrix $M$ invertible; further, the vectors are orthogonal (which will help us save some work).

Now, if we have sampled a signal in $2n$ locations / timepoints; then we can collect those samples in $\vec{f} \in \mathbb{R}^{2n}$.

If we can to express the signal as a linear combination of the cos/sin-vectors, all we have to do is solve the linear system

$$M[\vec{f}]_{\mathcal{B}} = \vec{f},$$

which in general requires roughly $\frac{8}{3} n^3$ operations ($\times/+$). This is the Super-Slow Fourier Transform.
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 $\mathbf{f}_B$, 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.
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 $1 - 1 = 0$.

---

### Comparing Operation Counts

<table>
<thead>
<tr>
<th>n</th>
<th>$4n^2$</th>
<th>$3n + n \log_2 n$</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>1,024</td>
<td>112</td>
<td>9</td>
</tr>
<tr>
<td>64</td>
<td>16,384</td>
<td>576</td>
<td>28</td>
</tr>
<tr>
<td>256</td>
<td>262,144</td>
<td>2,816</td>
<td>93</td>
</tr>
<tr>
<td>1,024</td>
<td>4,194,304</td>
<td>13,312</td>
<td>315</td>
</tr>
<tr>
<td>4,096</td>
<td>67,108,864</td>
<td>61,440</td>
<td>1,092</td>
</tr>
<tr>
<td>16,384</td>
<td>1,073,741,824</td>
<td>278,528</td>
<td>3,855</td>
</tr>
<tr>
<td>65,536</td>
<td>17,179,869,184</td>
<td>1,245,184</td>
<td>13,797</td>
</tr>
<tr>
<td>262,144</td>
<td>274,877,906,944</td>
<td>5,505,024</td>
<td>49,932</td>
</tr>
<tr>
<td>1,048,576</td>
<td>4,398,046,511,104</td>
<td>24,117,248</td>
<td>182,361</td>
</tr>
<tr>
<td>4,194,304</td>
<td>70,368,744,177,664</td>
<td>104,857,600</td>
<td>671,088</td>
</tr>
<tr>
<td>8,388,608</td>
<td>281,474,976,710,656</td>
<td>218,103,808</td>
<td>1,290,555</td>
</tr>
<tr>
<td>16,777,216</td>
<td>1,125,899,906,842,624</td>
<td>452,984,832</td>
<td>2,485,513</td>
</tr>
<tr>
<td>33,554,432</td>
<td>4,503,599,627,370,496</td>
<td>939,524,096</td>
<td>4,793,490</td>
</tr>
</tbody>
</table>

$33,554,432 = 2^{13} \times 2^{12} = 8,196 \times 4,096^\dagger$

$^\dagger$The “8k Digital Video Format” has a resolution of $8,192 \times 4,320$ pixels; the tentative Ultra High Definition Television (UHDTV) specification calls for $7,680 \times 4,320$ pixels for 16:9 aspect ratio (120 fps, 12 bits/channel (at least 3 – RGB) $\sim 4.0 \times 10^9$ 36-bit pixels/sec). IMAX shot on 70 mm film has a theoretical pixel resolution of $12,000 \times 8,700$ (at 24 fps, for a total of $2.5 \times 10^9$ pixels/sec).
<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>
Comparing Operation Counts

Fourier Transform: Computational Complexity

- FFT
- Slow FT
- Super Slow FT

Number of Operations

Number of Data Points: $n$

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩

Orthogonal Projections and Orthonormal Bases — (30/52)
Comparing Operation Counts

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

- FFT
- Slow FT
- Super Slow FT

Computational Time (seconds)

Number of Data Points: $n$

- Age of Earth
- One Lifetime
- One Year
- One Month
- One Day
- One Minute

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩
Available on Learning Glass videos:
5.1 — 5, 7, 10, 11, 15, 17, 27, 28
### Lecture – Book Roadmap

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Book, [GS5–]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>§4.1, §4.2, §4.4</td>
</tr>
<tr>
<td>5.2</td>
<td>§4.1, §4.2, §4.4</td>
</tr>
<tr>
<td>5.3</td>
<td>§4.1, §4.2, §4.4</td>
</tr>
</tbody>
</table>
### Metacognitive Exercise — Thinking About Thinking & Learning

<table>
<thead>
<tr>
<th></th>
<th>I know / learned</th>
<th>Almost there</th>
<th>Huh?!?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Right After Lecture</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>After Thinking / Office Hours / SI-session</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>After Reviewing for Midterm/Final</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(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 \( n \to \infty \).
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

$$W = \text{span} \left( \begin{bmatrix} 2 \\ 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 2 \\ 0 \\ 1 \end{bmatrix} \right), \quad \vec{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$
(5.1.28) Find the orthogonal projection of $\vec{e}_1$ onto the subspace $W$ of $\mathbb{R}^4$, where

$$W = \text{span} \left( \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \right), \quad \vec{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$
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.
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

\[
\begin{array}{c c l}
3 & \text{bytes/pixel} \\
\times & 24 & \text{frames/second} \\
\times & 7200 & \text{seconds} \\
\times & (1920 \times 1080) & \text{pixels/frame} \\
\hline
1,074,954,240,000 & \text{bytes}. \\
\end{array}
\]
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.
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...
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 $2n$ 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!
MPEG-4 Compression :: Enter the *Fourier Transform*

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

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

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-vector] coordinates (and back) using only dot products. What we have defined is known as the (one dimensional) *Fourier transform*.

**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 *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-vector] coordinates. Next, transform *that* “image” in the vertical direction. This now gives us an “image” of vertical [cos/sin-vector] coordinates of (horizontal [cos/sin-vector] of Gandalf). This is known as the two dimensional *Fourier transform*.
The two dimensional Fourier transform of Gandalf.
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%.
Gandalf, Fourier transformed and filtered: only the largest 4.6% of coefficients have been kept.
Gandalf, Fourier transformed and filtered: only the largest 4.6% of coefficients have been kept.
MPEG-4 Compression :: Wrap-up

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

References: