In the next few lectures, we will look at a few examples of orthobasis expansions that are used in modern signal processing.

Cosine transforms

The cosine-I transform is an alternative to Fourier series; it is an expansion in an orthobasis for functions on $[0, 1]$ (or any interval on the real line) where the basis functions look like sinusoids. There are two main differences that make it more attractive than Fourier series for certain applications:

1. the basis functions and the expansion coefficients are real-valued;
2. the basis functions have different symmetries.

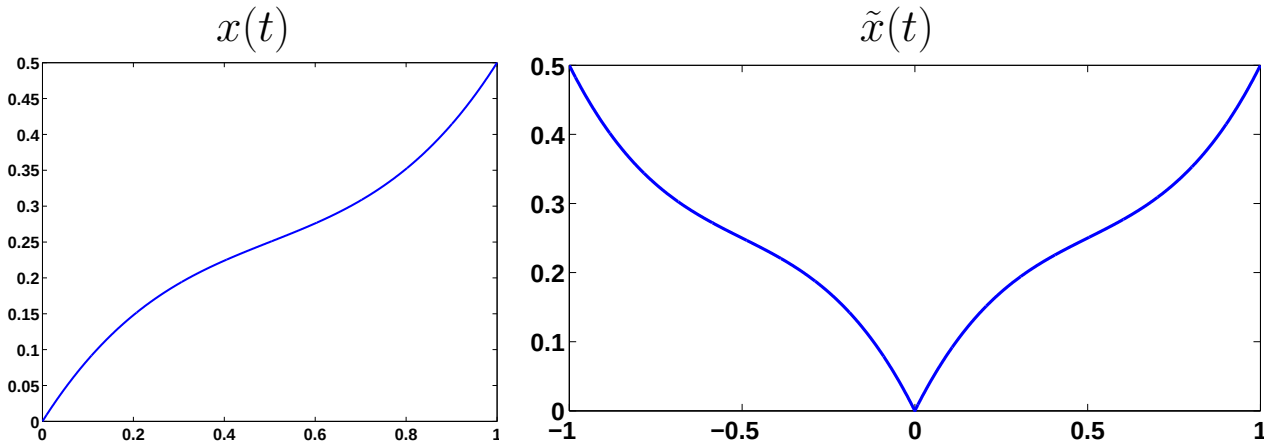
The discrete version of cosine-I (the "DCT") is used in both the JPEG image compression standard and the MPEG video compression standard; we will discuss this more later in this section.

Definition. The cosine-I basis functions for $t \in [0, 1]$ are

$$\psi_k(t) = \begin{cases} 1 & k = 0 \\ \sqrt{2}\cos(\pi k t) & k = 1, 2, \ldots \end{cases}. \tag{1}$$

We can derive the cosine-I basis from the Fourier series in the following manner. Let $x(t)$ be a signal on the interval $[0, 1]$. Let $\tilde{x}(t)$ be its "reflection extension" on $[-1, 1]$. That is

$$\tilde{x}(t) = \begin{cases} x(-t) & -1 \le t \le 0 \\ x(t) & 0 \le t \le 1 \end{cases}$$



We can use Fourier series to synthesis $\tilde{x}(t)$:

$$\tilde{x}(t) = \sum_{k=-\infty}^{\infty} \alpha_k \, e^{j\pi kt}.$$

Since $\tilde{x}(t)$ is real, we will have $\alpha_{-k} = \overline{\alpha_k}$, and so we can rewrite this as

$$\tilde{x}(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(\pi kt) + \sum_{k=1}^{\infty} b_k \sin(\pi kt),$$

where $a_0 = \alpha_0$, $a_k = 2\,\mathrm{Re}\,\{\alpha_k\}$, and $b_k = -2\,\mathrm{Im}\,\{\alpha_k\}$. Since $\tilde{x}(t)$ is even and $\sin(\pi kt)$ is odd, $\langle \tilde{x}(t), \sin(\pi kt) \rangle = 0$ and so

$$b_k = 0, \quad \text{for all } k = 1, 2, 3, \ldots,$$

and so $\tilde{x}(t)$ on $[-1, 1]$ can be written as

$$\tilde{x}(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(\pi kt).$$

Since we can use this expansion to build up **any** symmetric function on $[-1, 1]$, it means that the right hand side of the function on $[0, 1]$

is arbitrary, so **any** $x(t)$ on $[0,1]$ can be written as

$$x(t) = a_0 + \sum_{k=1}^{\infty} a_k \cos(\pi k t).$$

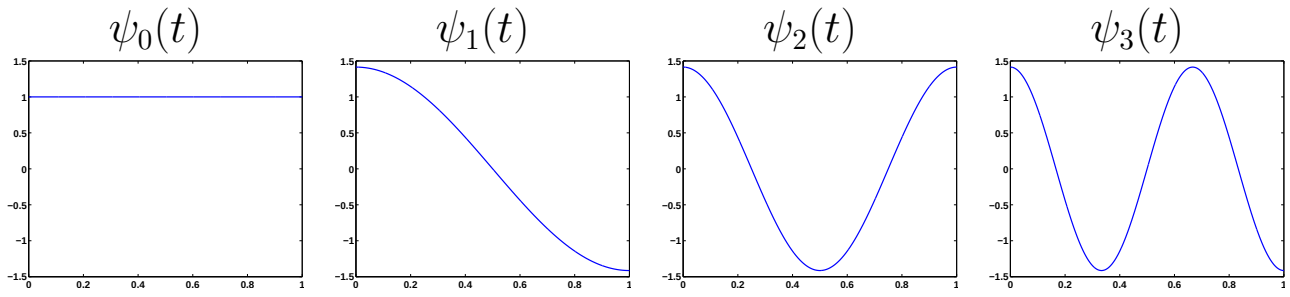
All that remains to show that $\{\psi_k : k = 1, 2, \ldots\}$ is an orthobasis is

$$\langle \boldsymbol{\psi}_k, \boldsymbol{\psi}_\ell \rangle = 2 \int_0^1 \cos(\pi k t) \cos(\pi \ell t) \, \mathrm{d}t = \begin{cases} 1 & k = \ell \\ 0 & k \neq \ell \end{cases}.$$

I will let you do this at home.

One way to think about the cosine-I expansion is that we are taking an **oversampled** Fourier series, with frequencies spaced at multiples of π rather than 2π, but then only using the real part.

Here are the first four cosine-I basis functions:



The discrete cosine transform (DCT)

Just as there is a version of Fourier series for **sampled** signals on an interval (i.e. finite dimensional signals in $\mathbb{C}^N$), this is the discrete Fourier transform (DFT), there is a version of the cosine-I transform for real-valued finite signals as well. This is called the **discrete cosine transform**, or DCT.

The DCT basis functions for $\mathbb{R}^N$ are

$$\psi_k[n] = \begin{cases} \sqrt{\frac{1}{N}} & k = 0 \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi k}{N}(n+\frac{1}{2})\right) & k = 1, \ldots, N-1 \end{cases} \tag{2}$$

for sample indices $n = 0, 1, \ldots, N-1$. Showing that

$$\sum_{n=0}^{N-1} \psi_k[n]\psi_\ell[n] = \begin{cases} 1 & k = \ell \\ 0 & k \neq \ell \end{cases}$$

is an exercise you can do at home. Notice that the samples of the cosines are on the half-sample points (we see $(n+1/2)$ in the expression above instead of n).

Just as the cosine-I transform can be computed from the Fourier series coefficients of a symmetric extension of the signal, the DCT can be computed from the DFT of a symmetric extension. That means we have a **fast algorithm** for computing the DCT — the cost is essentially the same as for an FFT, $O(N \log N)$.

The cosine-I and DCT for 2D images

Just as for Fourier series and the discrete Fourier transform, we can leverage the 1D cosine-I basis and the DCT into separable bases for 2D images.

Definition. Let $\{\psi_k(t)\}_{k\geq 0}$ be the cosine-I basis in (1). Set

$$\psi^{\text{2D}}_{k_1,k_2}(s,t) = \psi_{k_1}(s)\psi_{k_2}(t).$$

Then $\{\psi^{\text{2D}}_{k_1,k_2}(s,t)\}_{k_1,k_2\in\mathbb{N}}$ is an orthonormal basis for $L_2([0,1]^2)$

This is just a particular instance of a general fact. It is straightforward to argue (you can do so at home) that if $\{\psi_\gamma(t)\}_{\gamma\in\Gamma}$ is an orthonormal basis for $L_2([0,1])$, then $\{\psi_{\gamma_1}(s)\psi_{\gamma_2}(t)\}_{\gamma_1,\gamma_2\in\Gamma}$ is an orthonormal basis for $L_2([0,1]^2)$.

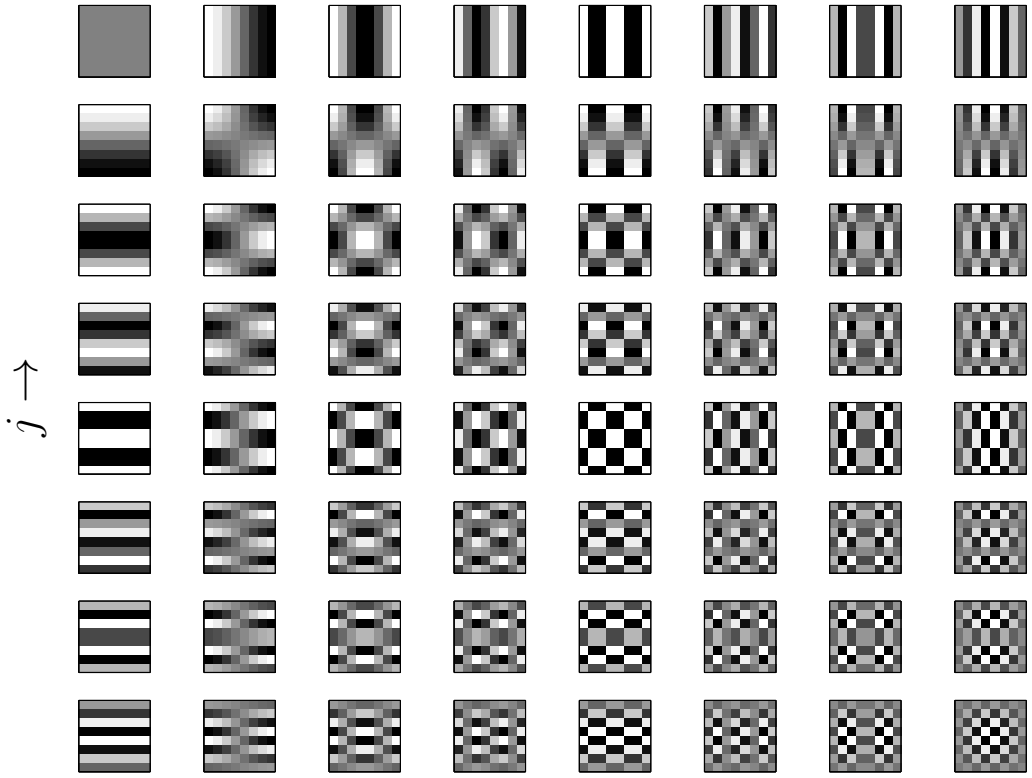
The DCT extends to 2D in the same way.

Definition. Let $\{\psi_k[n]\}_{0\leq k\leq N-1}$ be the DCT basis in (2). Set

$$\psi^{\text{2D}}_{j,k}[m,n] = \psi_j[m]\psi_k[n].$$

Then $\{\psi^{\text{2D}}_{j,k}[m,n]\}_{0\leq j,k\leq N-1}$ is an orthonormal basis for $\mathbb{R}^N \times \mathbb{R}^N$.

The 64 DCT basis functions for $N = 8$ are shown below:



$\psi_{j,k}[m, n]$ for $j, k = 0, \ldots, 7$

2D DCT coefficients are indexed by two integers, and so are naturally arranged on a grid as well:

$$\begin{bmatrix} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,N-1} \\ \alpha_{1,0} & \alpha_{1,1} & \cdots & \alpha_{1,N-1} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{N-1,0} & \alpha_{N-1,1} & \cdots & \alpha_{N-1,N-1} \end{bmatrix}$$

The DCT in image and video compression

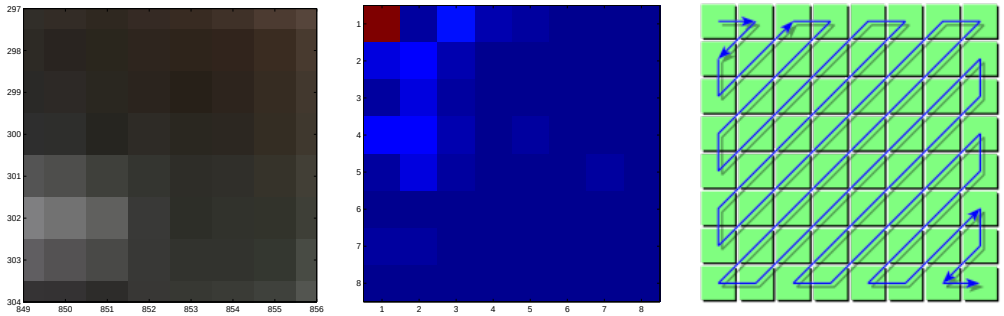
The DCT is basis of the popular JPEG image compression standard. The central idea is that while energy in a picture is distributed more or less evenly throughout, in the DCT transform domain it tends to be *concentrated* at low frequencies.

JPEG compression work roughly as follows:

1. Divide the image into 8×8 blocks of pixels
2. Take a DCT within each block
3. Quantize the coefficients — the rough effect of this is to keep the larger coefficients and remove the samller ones
4. Bitstream (losslessly) encode the result.

There are some details we are leaving out here, probably the most important of which is how the three different color bands are dealt with, but the above outlines the essential ideas.

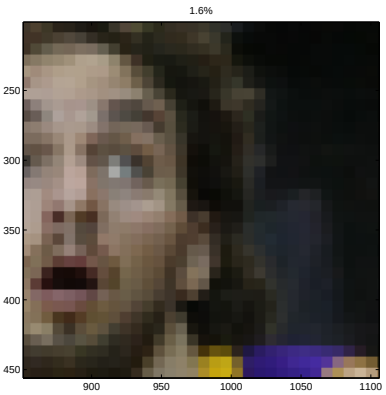
The basic idea is that while the energy within an 8×8 block of pixels tends to be more or less evenly distributed, the DCT concentrates this energy onto a relatively small number of transform coefficients. Moreover, the significant coefficients tend to be at the same place in the transform domain (low spatial frequencies).



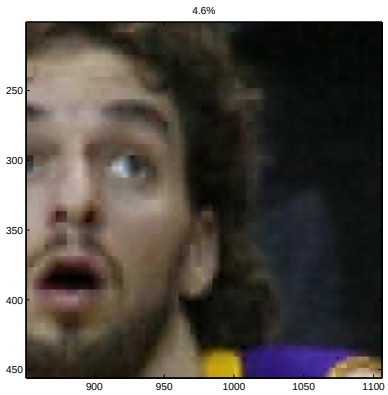
8×8 block 2D DCT coeffs ordering

To get a rough feel for how closely this model matches reality, let's look at a simple example. Here we have an original image 2048×2048, and a zoom into a 256×256 piece of the image:

Here is the same piece after using 1 of the 64 coefficients per block ($1/64 \approx 1.6\%$), $3/64 \approx 4.6\%$ of the coefficients, and $10/64 \approx 15.62\%$:



1/64



3/64

10/64

So the "low frequency" heuristic appears to be a good one.

JPEG does not just "keep or kill" coefficients in this manner, it quantizes them using a fixed quantization mask. Here is a common example:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

The quantization simply maps $\alpha_{j,k} \to \tilde{\alpha}_{j,k}$ using

$$\tilde{\alpha}_{j,k} = Q_{j,k} \cdot \text{round}\left(\frac{\alpha_{j,k}}{Q_{j,k}}\right)$$

You can see that the coefficients at low frequencies (upper left) are being treated much more gently than those at higher frequencies (lower right).

The **decoder** simply reconstructs each 8×8 block $\boldsymbol{x}_b$ using the synthesis formula

$$\tilde{\boldsymbol{x}}_b[m,n] = \sum_{k=0}^{7} \sum_{\ell=0}^{7} \tilde{\alpha}_{k,\ell}\, \phi_{k,\ell}[m,n]$$

By the Parseval theorem, we know exactly what the effect of quantizing each coefficient is going to be on the total error, as

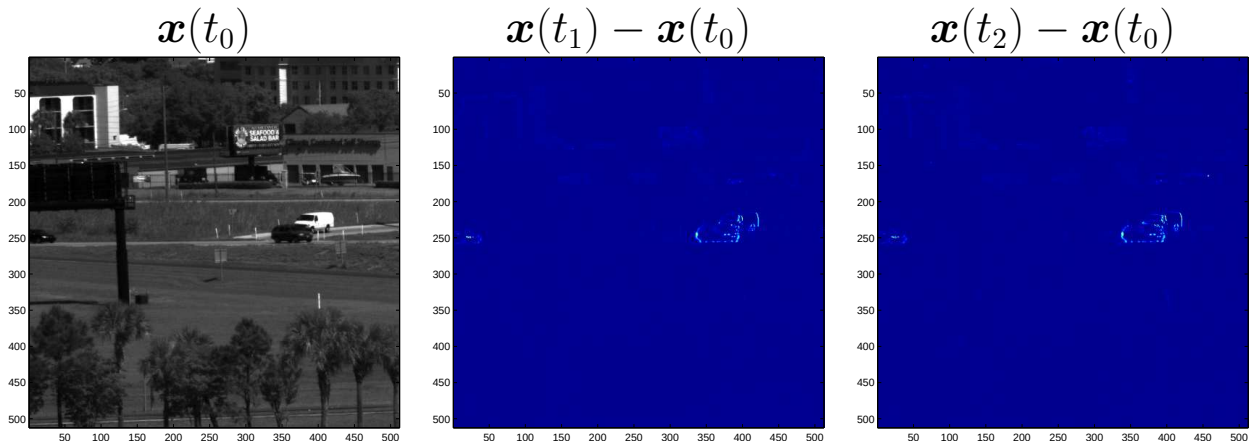
$$\|\boldsymbol{x}_b - \tilde{\boldsymbol{x}}_b\|_2^2 = \|\boldsymbol{\alpha} - \tilde{\boldsymbol{\alpha}}\|_2^2 = \sum_{k=0}^{7} \sum_{\ell=0}^{7} |\alpha_{k,\ell} - \tilde{\alpha}_{k,\ell}|^2.$$

Video compression

The DCT also plays a fundamental role in video compression (e.g. MPEG, H.264, etc.), but in a slightly different way. Video codecs are complicated, but here is essentially what they do:

1. Estimate, describe, and quantize the motion in between frames.
2. Use the motion estimate to "predict" the next frame.
3. Use the (block-based) DCT to code the residual.

Here is an example video frame, along with the differences between this frame and the next two frames (in false color):



The only activity is where the car is moving from left to right.