Applications of matrix computations Exercise 4, October 3–7, 2016

Lecturer: Samuli Siltanen.
Teaching assistants: Santeri Kaupinmäki and Jonatan Lehtonen.

Please send your solutions to application.matrixcomputation@gmail.com by Monday, October 10, at 10 AM.

1. Define three sets of vectors as follows:

$$\left\{ \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1000 \\ -1 \end{bmatrix} \right\}, \qquad \left\{ \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 3 \\ -3 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}, \qquad \left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1000 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right\}.$$

   (a) Use Matlab to determine which of the above sets are linearly independent.

   (b) Look up from the literature what is Gram-Schmidt orthogonalization. Explain it briefly (at most one page of text, preferably less).

   (c) Use Matlab and Gram-Schmidt orthogonalization to produce orthonormal bases out of the sets of vectors above that you found to be linearly independent.

2. Previously we approximated f with the real-valued formulation of Fourier series:

$$f(x) \approx a_0 + \sum_{n=1}^{N} \left(a_n \cos(nx) + b_n \sin(nx) \right), \tag{1}$$

   where $a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x) dx$, and for $n > 0$ we have $a_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(nx) dx$ and $b_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(nx) dx$. Now we have another, complex-valued formulation:

$$f(x) \approx \sum_{n=-N}^{N} c_n \, \varphi_n(x) \tag{2}$$

   with $\varphi_n(x) = (2\pi)^{-1/2} e^{inx}$ and

$$c_n = \langle f, \varphi_n \rangle = \int_0^{2\pi} f(x) \overline{\varphi_n(x)} dx \quad \text{for all } n \in \mathbb{Z}. \tag{3}$$

   (a) Show that $\langle \varphi_n, \varphi_m \rangle = 0$ when the integers n and m are not equal. Hint: You can use the results from previous exercises concerning $\sin(nx)$ and $\cos(nx)$.

   (b) Show that $\langle \varphi_n, \varphi_n \rangle = 1$.

   (c) Write a_n and b_n in terms of c_n.

   Note that parts a and b above were changed on Thursday to use φ_n.

   Some useful identities:

$$e^{\pm i\theta} = \cos(\theta) \pm i \sin(\theta), \qquad \overline{e^{i\theta}} = e^{-i\theta},$$
$$\cos(\theta) = \frac{1}{2}(e^{i\theta} + e^{-i\theta}), \qquad \sin(\theta) = \frac{1}{2i}(e^{i\theta} - e^{-i\theta}).$$

3. **Reducing Gibbs phenomenon** using a filtering trick related to the infamous Fejér kernel. The file *Fourierseries_complex_test.m* from Lecture 7 may be useful.

   (a) Use the complex Fourier series with a sequence of ever larger N, such as $N = 100, 200, 300, ...$, to approximate a function with jump discontinuities. Notice the rapid oscillations near the jumps (Gibbs phenomenon).

   (b) Apply the following filter to the coefficients: $c'_n = c_n(1 - |n|/N)$. What do you observe when you approximate $f(x)$ with

$$\sum_{n=-N}^{N} c'_n \, \varphi_n(x)?$$

4. Study the literature to understand Fast Fourier Transform for signals of length 2^m with $m > 1$.

   (a) What is the relation between c_n defined in (3) and the elements in the FFT vector? Hint: it is related to the "midpoint rule" numerical integration.

   (b) What is the trick that makes FFT fast? Explain it briefly (at most one page of text, preferably less).

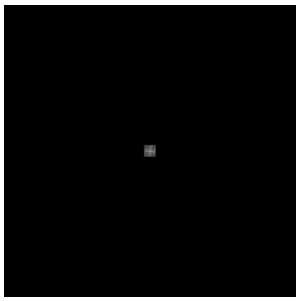
5. **Band-pass filtering of speech.** Download from the course website the Matlab file *Fourierseries_realsoundFFT_test.m* and use it as your starting point.

   (a) Record a short sentence using a laptop or a smartphone. Read the sound file into Matlab, truncate it to a power-of-two length 2^m and play it with the command `sound`.

   (b) Band-pass filter the signal using the index vector
   `index = (abs(nvec)<F1)|(abs(nvec)>F2);`
   Experiment with choices of the cutoff frequencies `F1` and `F2`.

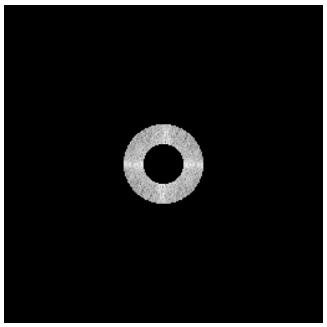
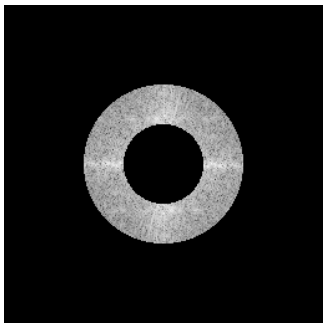
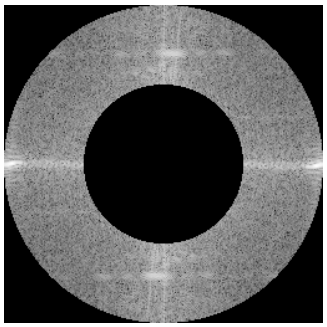
   (c) Adjust the cutoff frequencies `F1` and `F2` so that they are as close to each other as possible but you can still understand the sentence. How small can you make the compression ratio $(F_2 - F_1)/2^m$?

6. **Band-pass filtering of images.** Download the image file *BigBen.jpg* and the Matlab routine `FFTdemo2D.m`. It produces, for example, a low-pass filtered image shown below on the right. In the middle is shown the Fourier transform, or FFT, of the image with high frequencies put to zero.


   (a) Modify the file `FFTdemo2D.m` so that you get a disc-shaped low-pass filter instead of a square.

   (b) Further modify the file `FFTdemo2D.m` so that you get the following "band-pass" filters in the frequency domain.

   (c) Choose some small area in the image with a lot of detail, for example somewhere in the face of the clock. Show, side by side, that area of the original image, the low-pass filtered image of (a) and all the band-passed images of (b).