Applications of matrix computations, Spring 2018 Exercise 4

Lecturer: Samuli Siltanen.
Teaching assistants: Vesa Kaarnioja and Jesse Railo.

Deadline: Monday 23.04. at 10.00. *Sending your solutions late is not allowed!*

The exercises are held on Wed 14.15–16.00 (group 1, Exactum C321), Thu 12.15–14.00 (group 2, Exactum C321), Fri 10.15–12.00 (group 3, Exactum C128), and Fri 14.15–16.00 (group 4, Exactum C321). You gain credits from this course only by completing the weekly exercises. Participating in the exercise group is not obligatory for completing the course, but it is highly recommended.

Return your solutions as a single PDF file containing your Matlab codes (as text), results, images, comments, and explanations at the Moodle page of the course before the deadline (further instructions are given at the exercise groups if needed). Please include your name, student number, and your exercise group number in the beginning of the PDF file.

1. Let us evaluate the integrals needed in the derivation of Fourier series coefficients. (You can e.g. include a clear picture containing your calculations into your final pdf. Similarly in Problem 2 and (a) of Problem 5.)

   (a) Note the following trigonometric identities:

$$\sin x \sin y = \frac{1}{2}[-\cos(x+y) + \cos(x-y)], \tag{1}$$

$$\cos x \cos y = \frac{1}{2}[\cos(x+y) + \cos(x-y)], \tag{2}$$

$$\sin x \cos y = \frac{1}{2}[\sin(x+y) + \sin(x-y)]. \tag{3}$$

   Using equations (1)–(3), show that for all integers n and m satisfying $n \geq 1$ and $m \geq 1$ and $m \neq n$ we have

$$\begin{aligned} \int_0^{2\pi} \sin(nx)\sin(mx)dx &= 0, \\ \int_0^{2\pi} \sin(nx)\cos(mx)dx &= 0, \\ \int_0^{2\pi} \cos(nx)\cos(mx)dx &= 0. \end{aligned}$$

   (b) Show that for any $n \geq 1$ it holds that

$$\int_0^{2\pi} \sin(nx)\sin(nx)dx = \pi = \int_0^{2\pi} \cos(nx)\cos(nx)dx.$$

2. We can approximate f with the real-valued formulation of Fourier series:

$$f(x) \approx a_0 + \sum_{n=1}^{N} \left(a_n \cos(nx) + b_n \sin(nx) \right), \tag{4}$$

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:

$$f(x) \approx \sum_{n=-N}^{N} c_n \, \varphi_n(x) \tag{5}$$

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

$$c_n = \langle f, \varphi_n \rangle = \int_0^{2\pi} f(x) \overline{\varphi_n(x)} dx \quad \text{for all } n \in \mathbb{Z}. \tag{6}$$

(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 the previous exercise 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.

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. **Plotting pixel values of images.** Use the routine `PrepareImage.m` to extract a 1024×1024 grayscale image out of the color image *CladoniaFimbriana.jpg*.

(a) Plot the pixel values of the image along row 400. (Plot as a real function of one real variable in the interval $[0, 1]$.)

(b) As in (a), plot the pixel values along column 950.

(c) Plot the pixel values of the image along a circle centered in the middle of the image and having radius of roughly one fourth of the image width. Plot as a real function of one real variable in the interval $[0, 2\pi]$. (Hint: create x- and y-coordinates for the image pixels using `meshgrid`, and then use the routine `interp2` for picking out values.)

4. **Downsampling photographs.** Sometimes one needs to reduce the resolution of a digital image. A convenient way to do this is to take the average of $n \times n$ patches of pixels and use the result as the new pixel value. Then the new, downsampled image has size $(N/n) \times (M/n)$ if the original image has N rows and M columns. Here M and N need to be multiples of n.

   (a) Read the image file *BigBen.jpg* to Matlab. Apply the routine `MyDS2.m`, also available at the website, to downsample the image. (You also need the file `MyDScol.m`.) Repeat the downsampling two, three, and four times and observe the loss of detail in the image by plotting the downsampled versions of it.

   (b) Write a new routine called `MyDS3.m` that downsamples the input image using 3×3 patches instead of 2×2 patches as `MyDS2.m` does. Apply your new routine to the image *BigBen.jpg* after cropping it to a size that is divisible by three.

5. Let us use numerical quadrature for evaluating integrals of the form

$$\int_a^b f(x)dx,$$

   where $a < b$. We compare two numerical integration quadrature methods. In the lecture we discussed a simple Riemann sum approximation to integrals; it is sometimes called *midpoint rule*. That can be greatly improved by using the so-called *Simpson's rule*. For details you can see for example the Wikipedia page for Simpson's rule (see especially "Composite Simpson's rule" there).

   Evaluate the following integrals (a) analytically, (b) numerically using the midpoint rule, and (c) numerically using Simpson's rule:

$$\int_{-2}^{2} x^5 dx, \qquad \int_{-1}^{1} \sqrt{1-x^2}dx.$$

   Hint: note that the latter integral is the area of half the unit disc. Hint 2: whatever method you use, always start by making sure that integrating the constant function $f(x) \equiv 1$ results in $b - a$.

   If you use the same number of evaluation points for the midpoint rule and for Simpson's rule, which one gives the more accurate approximation? (To answer this question, you really need to implement Simpson's rule yourself and not use Matlab's adaptive `quad`, as it will choose the evaluation points automatically.)

6. This problem is about comparing midpoint rule (in our case more precisely left-point rule), trapezoidal rule, Simpson's rule and Gaussian quadrature in the context of numerical integration. Download the routines `quadrature_tests.m` and `gaussint.m` from the course website.

   The "left-point rule" for integrating over the integral is defined as follows. Fix n, set $h = 2\pi/n$ and define quadrature points $x_j \in [0, 2\pi]$ by the formula

   $$x_j := (j-1)h \qquad \text{for } j = 1, 2, 3, \ldots, n.$$

   Then approximate an integral by

   $$\int_0^{2\pi} f(x)dx \approx \sum_{j=1}^{n} hf(x_j) = h\sum_{j=1}^{n} f_j.$$

   (a) The following integral equals $\frac{8}{3}\pi^3$ (you don't need to prove this):

   $$\int_0^{2\pi} x^2 dx.$$

   Use n quadrature points with each of the three numerical integration methods for evaluating the above integral numerically. Note that the evaluation points are not necessarily the same for each method; however, make sure that the number of function evaluations is the same (in other words, equal to n for each method) in the comparison. Compute the relative error of each method by comparing to the analytical result. Repeat the computation with different values of n, for example $n = 10, 20, 30, \ldots$. Which of the methods converges the fastest? How many points do you need with the fastest method to achieve four significant digits correctly?

   (b) The following integral equals $2\sqrt{2\pi}$ (you don't need to prove this):

   $$\int_0^{2\pi} (2\pi - x)^{-1/2} dx.$$

   The function $(2\pi - x)^{-1/2}$ has a singularity near the point $x = 2\pi$, so we cannot evaluate it there. Therefore, we only compare the "left-point rule" and Gaussian quadrature as those methods do not involve the evaluation of $f(2\pi)$. Repeat the accuracy tests of (a) in this case.

   (c) The following integral equals $\frac{1}{6}\pi$ (you don't need to prove this):

   $$\int_0^1 \frac{1}{\sqrt{4 - x^2}} dx.$$

   Compute 10 decimals of π by approximating this integral using your favorite numerical quadrature, and solving for π algebraically. Keep increasing n as long as the 10 first decimals do not change anymore.