Applications of matrix computations Exercise 2, September 19–23, 2016

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

Send your solutions to application.matrixcomputation@gmail.com by Monday 26.09. at 10 AM.

1. Consider the following harmonic inpainting problem with six pixels:

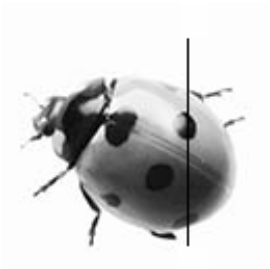
	100	100	
20	x_1	x_4	0
20	x_2	x_5	0
20	x_3	x_6	0
	100	100	

   (a) Write down the six linear equations describing the average equations.

   (b) Form a 6×6 matrix problem $Ax = b$ out of the linear equations.

   (c) Check that your matrix matches with that coming our of the routine `FD_Laplace.m` available at the course website. Read through the code in `FD_Laplace.m` and understand how it works.

2. Download the grayscale images *Ladybug.png* and *Ladybug2.png*. They look like this:

   There are one hundred pixels (in rows 18–117) missing in column 90 of *Ladybug2.png*, seen as a vertical black stripe. Use harmonic inpainting to fill the column, and compare the result to *Ladybug.png*. In other words,

   - call the missing pixel values $x_1, x_2, \ldots, x_{100}$,
   - organize them in a vertical vector $x = [x_1, x_2, \ldots, x_{100}]^T \in \mathbb{R}^{100}$,
   - write a system $Ax = b$ of linear equations requiring that each pixel x_j has average value of its four nearest neighbors (it is a good idea to initialize A as a sparse matrix with the command `A=sparse(100,100)`),
   - solve the 100×100 system using the Matlab command `x = A\b`, and finally
   - insert the inpainted pixel values into the image.

   Note that you can use the knowledge of the locations of the pixels. You do not have to find the "missing" pixels yourself.

3. Iterative solution of linear equations.

   (a) Let M be the 5×5 matrix constructed in the farm animal exercise of the first week. Write a Matlab function (defined in a separate m-file) called `M_mult.m` that takes a vector $x \in \mathbb{R}^5$ as input and produces the vector $Mx \in \mathbb{R}^5$ as output. Hint: learn how to use the Matlab command `function`.

   (b) Solve the farm animal problem in Matlab in the following three ways:
   ```
   x = inv(M)*b
   x = M\b
   x = gmres('M_mult',b)
   ```
   Compare the results. Note carefully that the `gmres` routine does not have the matrix `M` available, it can only evaluate `M*x(:)` for a given vector `x`.

   (c) Describe some situation in technology or computational science when iterative solution of an equation $Mx = b$ is needed. In such situation it is not practical to form the matrix M, but instead one can assume to have a routine taking a vector $x \in \mathbb{R}^n$ as input and producing the vector $Mx \in \mathbb{R}^n$ as output. This is called *matrix-free computation.* Hint: look for situations with large values of n.

4. Study the code in `Poisson_FD_solve.m` available at the course website. Take your own photo with your camera or smartphone, or use the picture *Smurf.jpg*.

   (a) Inpaint a small rectangle from one of the color channels of your photo using harmonic inpainting. Take away the downsampling and upsampling steps from the code; choose the rectangle so small that you can solve the full linear system with the computer you use. Present the result as a grayscale image.

   (b) Repeat the above with all color channels and present a color result.

5. 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) As verified in the lecture, the following integral equals $\frac{8}{3}\pi^3$:

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

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) As verified in the lecture, the following integral equals $2\sqrt{2\pi}$:

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

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

6. Let us evaluate the integrals needed in the derivation of Fourier series coefficients.

(a) Note the following trigonometric identities:

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

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

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

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

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

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

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