Numerical Methods for Differential Equations
Homework 4

Due by 2pm Tues 15 Nov. ***Please submit your hardcopy at the start of lecture.*** *The answers you submit should be attractive, brief, complete, and should include program listings and plots where appropriate. Possible approaches include "publish" in Matlab/Octave and* `pims.jupyter.ca`.

Introduction A digital image is usually represented using an rectangular array of pixels. In a gray-scale image, each pixel has an intensity value in $[0, 1]$ where 0 is black and 1 is white (often in practice, these values are "quantized" into integers in $[0, 255]$). The human eye is very sensitive to edges in an image. We're going to look at "sharpening" an image, by making the edges more obvious; this can make the image appear to be higher resolution than it actually is (technically this is known as "acutance"). This is, at best, the sort of thing televisions do when they claim to "upscale" low-rez signals to full HD or 4K.[1]

Given an image u, an "unsharp mask" first computes a diffused or blurry copy of u. This blurred copy is the subtracted from u to give an edge map. The edge map should be zero in regions where the image is smooth. Finally the edge map is added to the original image.

Problem 1: Image processing and Gaussian blurring

1. Download the file `hw4_images.zip` from the course website. Run the example code given.
2. Write a code to "blur" the image by using it as the initial condition for the heat equation $u_t = \nabla^2 u = u_{xx} + u_{yy}$. Use $h = 1$ (one pixel) and a time-step of $k = 0.1$ *with forward Euler time-stepping.* Use 10 steps.

   Update: You can use either periodic or zero-neumann boundary conditions *as long as you clearly state which you use.* The former is easier but not really physically appropriate for many images; the latter (zero derivative in the normal direction, $\frac{\partial u}{\partial n} = 0$) should give better results.
3. Produce a close-up picture showing the result of `testpat_noblur.png`.
4. Produce another picture using $k = 0.5$: WTF just happened and why?

Problem 2: The Unsharp Mask

1. Modify your code to perform an unsharp mask based on the algorithm described above. You can use the images in the `.zip` file to test.
2. Using parameter values as shown above, display the results (zoomed in) for `testpar_blur2.png`. What are the minimum and maximum values of u (i.e., the lightest and darkest pixel values for the result).
3. If the image is not blurry to begin with (e.g., `textpat_noblur.png`), what does the unsharp mask do locally around edges? Your solution should show a "zoom-in" of the results. The number of blurring steps is a tuneable parameter in this algorithm: what effect does it have?

The tutorial at `http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm` gives some more details and also discusses the biological reasons of why this works. It also cautions photographers, quite rightly, against overusing this technique.

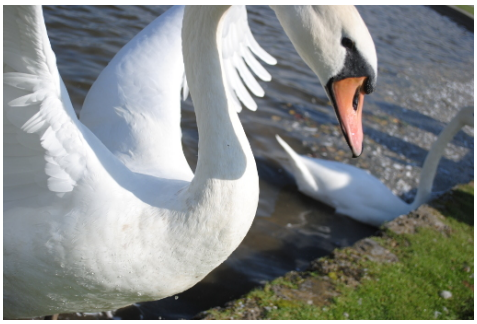
[1] Caution, learning too much about digital image processing can really spoil television for you! Ingrid Daubechies (of "Wavelets" fame) tells a story of watching football and recognizing from the compression artifacts that the broadcaster was (over-)using one of the wavelet techniques that she invented.

Problem 3 (Perona–Malik Image denoising) Let's look at a fully nonlinear application in image processing. First try regular diffusion (the heat equation in 2D). This is Gaussian blurring. It should get rid of the noise but it will also blur all the detail in the image and in particular the *edges*. The "correct" boundary condition is probably zero-Neumann but for simplicity, you could start with periodic.

The Perona–Malik equation [2, 3] is a modification of Gaussian diffusion that employs edge preservation by varying the diffusion coefficient across the image, penalising it at edges. The general equation is

$$u_t = \nabla \cdot \big(g(\|\nabla u\|)\nabla u\big), \qquad \text{where} \qquad g(s) = \frac{1}{1+\frac{s^2}{\lambda^2}}$$

is an edge detection function with a tunable parameter λ that controls the sensitivity of the equation to visual edges. It gives a threshold to separate noise from edges.



Photograph by Harry Biddle [1]

The two-dimensional Cartesian form of the Perona–Malik equation is

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left[g\left(\sqrt{u_x^2+u_y^2}\right)u_x\right] + \frac{\partial}{\partial y}\left[g\left(\sqrt{u_x^2+u_y^2}\right)u_y\right].$$

Following the lecture notes, this can be discretized with finite differences and forward Euler

$$v_{ij}^{n+1} = v_{ij}^n + k\left[\frac{g_{i+\frac{1}{2},j}^n D_+^x v_{ij}^n - g_{i-\frac{1}{2},j}^n D_-^x v_{ij}^n}{h} + \frac{g_{i,j+\frac{1}{2}}^n D_+^y v_{ij}^n - g_{i,j-\frac{1}{2}}^n D_-^y v_{ij}^n}{h}\right]. \tag{1a}$$

Here D_+^α and D_-^α indicate forward and backward finite differences, in the direction indicated by the superscript α, on a grid spacing of h. The expressions involving g between grid points are calculated as the average of the values at the two neighbouring grid points, e.g.,

$$g_{i+\frac{1}{2},j}^n = \tfrac{1}{2}(g_{i+1,j}^n + g_{ij}^n), \qquad g_{i-\frac{1}{2},j}^n = \tfrac{1}{2}(g_{i-1,j}^n + g_{ij}^n), \tag{1b}$$

where the nodal g_{ij}^n are computed using central finite differences

$$g_{ij}^n = g\left(\sqrt{(D_c^x v_{ij}^n)^2 + (D_c^y v_{ij}^n)^2}\right). \tag{1c}$$

Download the code from the website which loads the image and adds noise. Write a code to perform Perona–Malik denoising. Assuming you choose to make $h_x = h_y = 1$, then you might start with $k = 0.05$ and $\lambda = 1/100$.

You might find the Matlab/Octave code "FDmatrices" useful. It is available at `https://gitlab.math.ubc.ca/cbm/fdmatrices` or `https://github.com/cbm755/fdmatrices`.

Problem 4: Git (for bonus points). If you haven't done your Git homework from previous HW, you still have a chance (see HW3 for problem).

References

[1] H. Biddle, I. von Glehn, C. B. Macdonald, and T. März. A volume-based method for denoising on curved surfaces. In *Proc. ICIP13, 20th IEEE International Conference on Image Processing*, pages 529–533, 2013.

[2] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. In *Proceedings of IEEE Computer Society Workshop on Computer Vision*, pages 16–22, 1987.

[3] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.