Mini-project #2

The goal of this mini-project is to study the use of the SVD for solving ill-posed problems.

We will consider the ill-posed problem of de-blurring an image. Suppose $f(x)$ is an original image ($x \in R^2$) defined in a square. A blurring operator $\mathcal{A}$ applied to f produces a blurred image $f_{\text{blurred}}(x)$:

$$f_{\text{blurred}}(x) = \mathcal{A}f(x).$$

Usually $\mathcal{A}$ is linear and can be modeled by a convolution integral (the latter fact is not important for us). For our purposes, f and f_{blurred} will be represented by their values on a Cartesian grid (i.e., by matrices), and $\mathcal{A}$ will be replaced by a symmetric real matrix A. To model a real life situation we will assume that the blurred image results form some sort of measurements, and, thus, is contaminated by noise. In other words, our data $g(x)$ are equal to $f_{\text{blurred}}(x) + \eta(x)$ where $\eta(x)$ is a random error. Our goal is to reconstruct (approximately) $f(x)$ from $g(x)$. The project will be done in a several steps. Steps 1 to 3 model the forward problem; Steps 4 to 8 will solve the inverse problem (de-blurring).

Step 1.

We need some sort of an image $f(x)$. Let us generate $f(x)$ as a sum of several characteristic functions of non-overlapping disks of different diameters. (Then, $f(x)$ at each point will take the value either 0 or 1). The exact choice of parameters is up to you. The image will be represented by the function $f(x)$ computed on a grid 65×65. Make a figure showing $f(x)$ as a gray scale image. (Use commands **imagesc(...); axis image, colormap('gray');**)

Step 2.

Download from my webpage function **makematrix.** (Read the description inside.) Given the image dimension (ndim=65) this function will return (4225×4225) real symmetric matrix A. In order to apply A to the image $f(x)$, the latter needs to be re-shaped as a long column vector X of size (4225×1). This can be done using Matlab function **reshape**. Compute vector $Y = AX$. Re-shape Y back to the image of size 65×65. This is your $f_{\text{blurred}}(x)$. Make a gray scale plot of it.

Step 3.

Using **randn,** generate a 65×65 random image $\eta(x)$. Scale it so that the 2-norm of $\eta(x)$ equals 10^{-4} times the 2-norm of $f_{\text{blurred}}(x)$. Form $g(x) \equiv f_{\text{blurred}}(x) + \eta(x)$. There is no need to submit the plot of $g(x)$ since it should look indistinguishable from $f_{\text{blurred}}(x)$. Image $g(x)$ models the solution of the forward problem, representing measurements of blurred and noisy image. The goal of the inverse problem is to find an approximation to $f(x)$ from given $g(x)$.

Step 4.

You may try for find $f_{\text{approx}}(x)$ by solving the problem

$$g(x) = \mathcal{A} f_{\text{approx}}(x). \tag{1}$$

(You need to remember that all such solutions involve multiple re-shaping of 2D images in 1D vectors and back). Convince yourself that the direct solution of the above system yields a very poor (to put it mildly) approximation.

Step 5.

Compute the SVD of A. (Use **svd).** What's the condition number of A relative to the 2-norm? (Do not use **cond).** Convince yourself that the use of SVD to solve (1) directly does not help either.

Step 6

Instead of trying to solve (1), consider a reduced-rank matrix, $A_{\text{regularized}}$ whose lower singular values are replaced by zeros. Let's call the cut-off parameter r; $r \in (0, 1)$. Singular values σ_j that are greater or equal to $r\sigma_1$ are kept; the rest of them is set to zero. Now one can solve the system

$$g(x) = \mathcal{A}_{\text{regularized}} f_{\text{approx}}(x) \tag{2}$$

for f_{approx} but only in the least square sense. (Do not form $\mathcal{A}_{\text{regularized}}$ explicitly!!!! Instead use U, V, and Σ from the original SVD of A to solve system (2)

Compute such approximations for values of $r = 0.0002, 0.0001, 0.00005$; make gray-scale plots of the corresponding images; compute the relative errors of reconstruction ($f_{\text{approx}}(x)$ compared to $f(x)$), in 2-norm (don't forget to print them out). Which of the reconstructions better according to the 2-norm?

Step 7

Experiment with different values of r. Find a value of r that gives a better reconstruction in 2-norm than those in Step 6. Make a plot, print the better value of r you found.

Step 8

Does the relative error in 2-norm coincide with visual quality of the reconstruction? In other words, if our goal is to recover sharp edges in the image, is the best reconstruction the one with the lowest error in the 2 norm?