You can't go home again

In a previous lab you learned to blur an image, represented as an $m \times n$ pixel intensity matrix $\boldsymbol{X}$, through multiplying by a matrix on each side. Let $\boldsymbol{V} = \boldsymbol{B}_m$ and $\boldsymbol{H} = \boldsymbol{B}_n^T$, where each $\boldsymbol{B}_p$ is a $p \times p$ local blurring matrix returned by your function `blurmatrix`. For some positive integer k,

$$\boldsymbol{Z} = \boldsymbol{V}^k \boldsymbol{X} \boldsymbol{H}^k \tag{1}$$

applies local blurring k times in each direction to get the resulting $\boldsymbol{Z}$.

It's much more common, of course, to have a blurred image that you want to restore, or *deblur*. We can easily express the process using matrix inverses:

$$\boldsymbol{X} = \boldsymbol{V}^{-k} \boldsymbol{Z} \boldsymbol{H}^{-k}.$$

However, in computational practice we rarely use matrix inverses; instead, we solve linear systems. In MATLAB this is done by using \ (backslash) when multiplying by an inverse on the left, and / (forward slash) when multiplying by an inverse on the right.

It's intuitively clear that blurring and deblurring are fundamentally different. While we can blur an image as much as we like, we expect that a severely blurred image cannot be accurately restored. In finite precision, this proves to be the case. We will soon be learning how to characterize matrix multiplications like this that cannot easily be undone.

Goals

You will blur an image as before, then apply backslash and slash to deblur it. For a small amount of blur this will succeed, but not when you try to restore a more severely blurred image.

Preparation

Read Section 2.3 and review the lab on blurring an image. You will need a working copy of `blurmatrix.m`.

Procedure

Download the script template and make `blurmatrix.m` available.

1. Using `blurmatrix`, let $\boldsymbol{V} = \boldsymbol{B}_{100}$. Make side-by-side plots of $\boldsymbol{V}$ and $\boldsymbol{V}^{-1}$ using `imagesc` (see the template). Note that while the nonzeros of $\boldsymbol{V}$ are all near the diagonal, the same is not true for $\boldsymbol{V}^{-1}$, so deblurring is not a simple local operation.

2. Download and locally save an image. For best results, use a cartoon or line drawing of size 500 pixels or less on each side. Import the image and call the result $\boldsymbol{X}$, the original image.

3. Make side-by-side plots of $(V^{-1})^6$ and $(V^6)^{-1}$. These should be equivalent, but your results will not be—our first sign of some trouble.

4. Using $k = 1$, let $\boldsymbol{Z}$ be the blurred image as described in equation (1) . Using `subplot` and `imshow`, make side by side pictures of the original and blurred images.

5. Now let $\boldsymbol{Y}$ be the result of deblurring $\boldsymbol{Z}$. Use \ and /, as described in the introduction. Make side by side pictures of $\boldsymbol{X}$ and $\boldsymbol{Y}$. You should see apparently identical images.

6. Repeat steps 4 and 5, but using $k = 6$. The restoration results should be *very* different!