Project: Deblur, denoise, delight!

Recall that an image may be represented as an $m \times n$ matrix $\boldsymbol{X}$ of pixel intensities. A fair way to simulate blurring the image is by

$$\boldsymbol{Z} = (\boldsymbol{B}_m)^p \boldsymbol{X} (\boldsymbol{B}_n^T)^p, \tag{1}$$

where p is a positive integer and each $\boldsymbol{B}_k$ is a $k \times k$ symmetric tridiagonal matrix with 1/2 all along the main diagonal, and 1/4 all along both the sub- and superdiagonals. As can easily be verified, the mapping from $\boldsymbol{X}$ to $\boldsymbol{Z}$ is linear, so if we represent the images as vectors (say, by stacking columns), then $\boldsymbol{z} = \boldsymbol{G}\boldsymbol{x}$. The matrix $\boldsymbol{G}$ must be $mn \times mn$, and can be found explicitly, but we consider it to be unavailable.

Suppose that the image $\boldsymbol{Z}$ is derived from both blurring of $\boldsymbol{X}$ and some added noise. We can write this as

$$\boldsymbol{z} = \boldsymbol{G}\boldsymbol{x} + \boldsymbol{y}, \quad \text{for some } \boldsymbol{y} \text{ with } \|\boldsymbol{y}\|^2 = \delta^2, \tag{2}$$

where δ is a small(?) number. Our goal is the reconstruction problem: for known $\boldsymbol{z}$, $\boldsymbol{G}$, and δ, return the best (most likely) $\boldsymbol{x}$. An optimization is essential, since there are many possible solutions of (2).

Truly random noise will look like static, with lots of value jumps from pixel to pixel. Real images, on the other hand, usually have large patches of slowly-varying values. Consider the $k \times k$ matrix

$$\boldsymbol{R}_k = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}. \tag{3}$$

$\boldsymbol{R}_k$ is a scaled finite-difference approximation of the second derivative, so $\|\boldsymbol{R}_k\boldsymbol{u}\|$ is a measure of the "roughness" of the vector $\boldsymbol{u}$. The 2D equivalent of this process on an image matrix $\boldsymbol{X}$ is to apply roughening from both the left and the right, in the form

$$\boldsymbol{R}_m\boldsymbol{X} + \boldsymbol{X}\boldsymbol{R}_n^T. \tag{4}$$

This too is a linear transformation of the image $\boldsymbol{X}$, which we denote by $\boldsymbol{L}\boldsymbol{x}$ for an unspecified matrix $\boldsymbol{L}$. Minimizing roughness, as measured by $\|\boldsymbol{L}\boldsymbol{x}\|$, seems like a reasonable optimization criterion.

We now have a complete problem to state:

$$\min \|\boldsymbol{L}\boldsymbol{x}\|^2 \text{ over all } \boldsymbol{x} \text{ satisfying } \|\boldsymbol{G}\boldsymbol{x} - \boldsymbol{z}\| = \delta. \tag{5}$$

This is a multidimensional minimization subject to a scalar constraint function, a problem nicely changed to an unconstrained form by the method of Lagrange multipliers. Introducing the unknown scalar λ, we obtain the simultaneous equations

$$(\boldsymbol{L}^T\boldsymbol{L} + \lambda\boldsymbol{G}^T\boldsymbol{G})\boldsymbol{x} = \lambda\boldsymbol{G}\boldsymbol{z} \tag{6}$$

$$\|\boldsymbol{G}\boldsymbol{x} - \boldsymbol{z}\| - \delta = 0. \tag{7}$$

Equation (6) is an $mn \times mn$ linear system that defines $\boldsymbol{x}$ when λ is given. We could pose (7) as a rootfinding problem for λ. However, we rarely know a good value for δ in advance, so it makes more sense to regard the solution $\widehat{\boldsymbol{x}}$ of (6) as a function of a tunable parameter λ. As $\lambda \to 0$, greater emphasis is placed on minimizing roughness (increased smoothing), and as $\lambda \to \infty$, more emphasis is on deblurring (increased sharpness).

The matrices $\boldsymbol{L}$ and $\boldsymbol{G}$ are never needed for these computations. It's possible to show that $\boldsymbol{L}$ and $\boldsymbol{G}$ are both symmetric, so $\boldsymbol{L}^T\boldsymbol{L}\boldsymbol{x} = \boldsymbol{L}(\boldsymbol{L}\boldsymbol{x})$ and $\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{x} = \boldsymbol{G}(\boldsymbol{G}\boldsymbol{x})$. For implementation, the multiplication $\boldsymbol{L}\boldsymbol{x}$ or $\boldsymbol{G}\boldsymbol{x}$ is achieved by reshaping $\boldsymbol{x}$ to the image matrix $\boldsymbol{X}$, applying (4) or (1), respectively, and then reshaping back into a vector. Hence, given λ, the operator on the left side of (6) can be implemented by writing a function that can be passed to `gmres` in lieu of a matrix.

A final note: Adding a constant to all the entries of $\boldsymbol{Z}$ can improve the reconstruction. A typical choice is to make the mean value of the pixels equal to zero. This doesn't affect the plot using `imagesc`, which maps min and max pixel values to black and white.

Project assignment

Submit a zip archive with files as directed here. Don't forget to convert an imported image to a 2D matrix of class `double`. **You should use `imagesc` for plotting all images, and after each image plot use**

```
colormap(gray(256))), axis equal
```

Objective 1. Derive (6)–(7) from (5). You will need to consult a calculus book, and to take the gradients of expressions like $\boldsymbol{C}^T\boldsymbol{C}\boldsymbol{u}$ and $\boldsymbol{u}^T\boldsymbol{c}$ with respect to $\boldsymbol{u}$.

For this objective, submit the following:

- A PDF, `objective1.pdf`, typeset nicely or scanned from a neatly written document and showing the derivation in convincing detail.

Objective 2. Write two functions:

```
function Z = blur(X,p)
function Z = roughen(X)
```

They implement the equivalent of $\boldsymbol{G}\boldsymbol{x}$ and $\boldsymbol{L}\boldsymbol{x}$, respectively, but by operating on $m \times n$ images via (1) and (4). In practice, both m and n are small enough that it's not important to use sparse matrices. You can test your functions on the image `checkeredflag` available on the website.

For this objective, submit the following:

- Files `blur.m` and `rough.m`. Each should be self-contained.
- Two PNG images. One shows the original and blurred (with $p = 2$) checkered flag images side by side, and the other shows the original and roughened images side by side.

Objective 3. Write a function

```
function y = linop(x,lambda,p,m,n)
```

The inputs are a vectorized $mn \times 1$ image $\boldsymbol{x}$, a positive scalar value for λ, and the size $m \times n$ of the original image. The output is $\boldsymbol{y} = (\boldsymbol{L}^T\boldsymbol{L} + \lambda\boldsymbol{G}^T\boldsymbol{G})\boldsymbol{x}$, computed using calls to `blur` and `roughen`.

For this objective, submit the following:

- The file `linop.m`. It should depend only on native MATLAB commands and `blur` and `roughen`.
- A PNG file showing the result of applying `linop` to the checkered flag image with $\lambda = 8$, $p = 4$.

Objective 4. Download `fonzie.png` from the website and import it as the degraded image $\boldsymbol{Z}$. For $p = 3$ and $\lambda = 0.05, 0.2, 1, 5$, apply `gmres` using `linop` to reconstruct the image $\widehat{\boldsymbol{X}}$ (the matrix form of the solution of (6)).

For this objective, submit the following:

- Four PNG images, each showing one of your reconstructions with the value of λ in the title.

Objective 5. Download `plate.png` from the website and import it as the degraded image $\boldsymbol{Z}$. It's a severely degraded image of a license plate. Experiment with p and λ and do your best to identify all of the characters on the plate. You might want to use different parameter values for different characters.

For this objective, submit the following:

- One or more PNG images, showing your best reconstructions for the characters on the plate. The title of each image should show the values of p and λ used to generate it.