Project: That's one thing he hated! The NOISE! NOISE! NOISE! NOISE!

Most digital images are captured by a photoelectric sensor array. If the array is divided into many small pixels, then the number of photons gathered by each pixel may be small, and random fluctuations can cause "noise" in the image. For our purposes we model noise by the addition at each image array element of a normally distributed random number with average zero. Hence for an $n \times n$ image,

```
Z = X + sigma*randn(n)/n;
```

There is no perfect way to remove noise once it has been added. One solution is to apply a good dose of blurring, but this tends to remove features of interest as well as noise. Instead there is a method called constrained least squares inverse filtering. In this model we have to assume knowledge about the strength σ of noise added. If we write the noisy image $\boldsymbol{Z}$ as a vector $\boldsymbol{z}$, we can describe the result of the process as the $\boldsymbol{u}$ satisfying

$$\min \|\boldsymbol{R}\boldsymbol{u}\|, \quad \text{subject to} \quad \|\boldsymbol{u} - \boldsymbol{z}\| = \sigma. \tag{1}$$

Here $\boldsymbol{R}$ is a "roughing" operator. It is meant to exaggerate and thus penalize discontinuities. As is often the case, we find it convenient to describe and compute the action of $\boldsymbol{R}$ on the matrix form $\boldsymbol{V}$ of an image vector $\boldsymbol{v}$:

$$\boldsymbol{R}\boldsymbol{v} \leftrightarrow \boldsymbol{D}\boldsymbol{V} + \boldsymbol{V}\boldsymbol{D}, \tag{2}$$

where $\boldsymbol{D}$ is the matrix

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

In words, (1) says the solution to the problem is the least rough image $\boldsymbol{u}$ that is the same distance from the noisy image as the original.

As a constrained minimization, equation (1) has a solution in terms of a Lagrange multiplier $\gamma > 0$. After some work, the solution is

$$\boldsymbol{u} = (\boldsymbol{I} + \gamma \boldsymbol{R}^2)^{-1}\boldsymbol{z}, \tag{4}$$

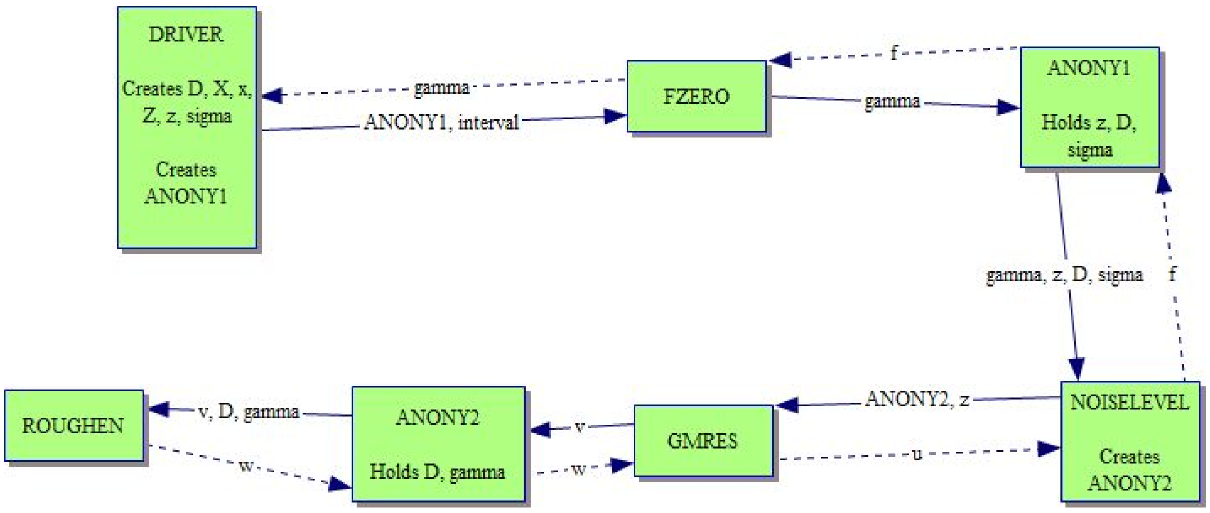
where γ is chosen so that

$$f(\gamma) = \|\boldsymbol{u} - \boldsymbol{z}\|^2 - \sigma^2 = 0. \tag{5}$$

This is a rootfinding problem for γ. Note that each evaluation of f requires the solution of a linear system of equations for $\boldsymbol{u}$. This system is large and will be solved using GMRES. However, you don't have the matrix $\boldsymbol{R}$, so you will have to use GMRES in its matrix-free mode. (Make sure you monitor the GMRES flag output so that returned values are really solutions.)

To make this project simpler, choose $\boldsymbol{X}$ to be a *square* image of size no larger than 250×250. If you use a photograph, start with a high-res image and crop to the smaller size to get a good image. Also use an image with a good combination of dark and light pixels, rather than mostly black or mostly white.

The structure of the code you need for this problem is shown in the figure. You need to write



- Function `roughen`, which returns $\boldsymbol{w} = (\boldsymbol{I} + \gamma \boldsymbol{R}^2)\boldsymbol{v} = \boldsymbol{v} + \gamma \boldsymbol{R}^2 \boldsymbol{v}$ for any given vector $\boldsymbol{v}$. This includes some reshaping, and applying (2) two times.

- Function `noiselevel`, which calculates f in equation (5) above after GMRES has been called to find $\boldsymbol{u}$.

- A top-level driver that defines the images, the value of `sigma`, and the matrix $\boldsymbol{D}$ needed by `roughen`. Once `fzero` has found the correct value of γ, you need to call GMRES one more time to find the $\boldsymbol{u}$ from (4). The result is the method's optimal reconstruction of $\boldsymbol{x}$, which is reshaped to find $\boldsymbol{X}$.

There are also two "wrapper" anonymous functions needed to pass extra values through `fzero` and `gmres`. These are one-liners like in the earlier project and the homework.

Your goal is to apply noise with `sigma` equal to $8n$, $16n$, and $32n$, and then apply reconstruction in each case. Report your γ solution each time (it should increase with σ and stay less than or at least close to 1), and show the noisy and reconstructed images side by side each time.