Homework #3

2019 Fall Optimization Theory and Application

National Chiao Tung University, Hsinchu 300, Taiwan

Due on 17:30, Dec. 13, 2019

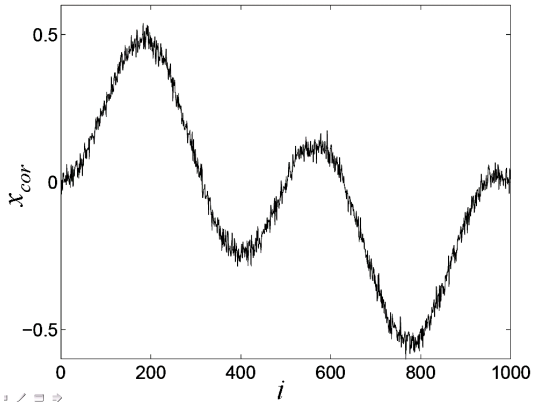
2019/12/6

Discussion is encouraged but should work out this homework independently.

No cheating in the homework.

You have to write your own Matlab®/C/C++/Python/…. codes to complete this homework.

1. **(100 pts)** The figure shows a signal of length 1000, corrupted with noise.



We want to estimate the original signal. This is called signal reconstruction, denoising, or smoothing. In this homework, we apply a smoothing method based on least squares. We will represent the corrupted signal as a vector x_{cor} of size 1000. (The values can be obtained as xcor = denoise using the file denoise.m.) The estimated signal (i.e., the variable in the problem) will be represented as a vector $\hat{x}$ of size 1000. The idea of the method is as follows. We assume that the noise in the signal is the small and rapidly varying component. To reconstruct the signal, we decompose x_{cor} in two parts $x_{cor} = \hat{x} + v$ where v is small and rapidly varying, and $\hat{x}$ is close to x_{cor} ($\hat{x} \approx x_{cor}$) and slowly varying ($\hat{x}_{i+1} \approx \hat{x}_i$). We can achieve such a decomposition by choosing $\hat{x}$ as the solution of the least squares problem

$$\text{Minimize}\left\{ \| x - x_{cor} \|^2 + \lambda \sum_{i=1}^{999} (x_{i+1} - x_i)^2 \right\} \ldots\ldots\ldots (1)$$

where λ is a positive constant. The first term $\|x - x_{cor}\|^2$ measures how much x deviates from x_{cor}. The second term, $\sum_{i=1}^{999} (x_{i+1} - x_i)^2$, penalizes rapid changes of the signal between two samples. By minimizing a weighted sum of both terms, we obtain an estimate $\hat{x}$ that is close to x_{cor} (i.e., has a small value of $\|x - x_{cor}\|^2$) and varies slowly (i.e., has a small value of $\sum_{i=1}^{999} (\hat{x}_{i+1} - \hat{x}_i)^2$). The parameter λ is used to adjust the relative weight

of both terms. The model (1) is a least squares problem, because it can be expressed as

$$\text{Minimize} \| Ax - b \|^2 \text{ .....(2)}$$

Where

$$A = \begin{bmatrix} I \\ \sqrt{\lambda} D \end{bmatrix}, \quad b = \begin{bmatrix} x_{cor} \\ 0 \end{bmatrix},$$

and D is a 999 × 1000 matrix defined as

$$D = \begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & -1 & 1 \end{bmatrix}.$$

The matrix A is quite large (1999 × 1000), but also very sparse, so we will solve the least squares problem by solving the normal equations

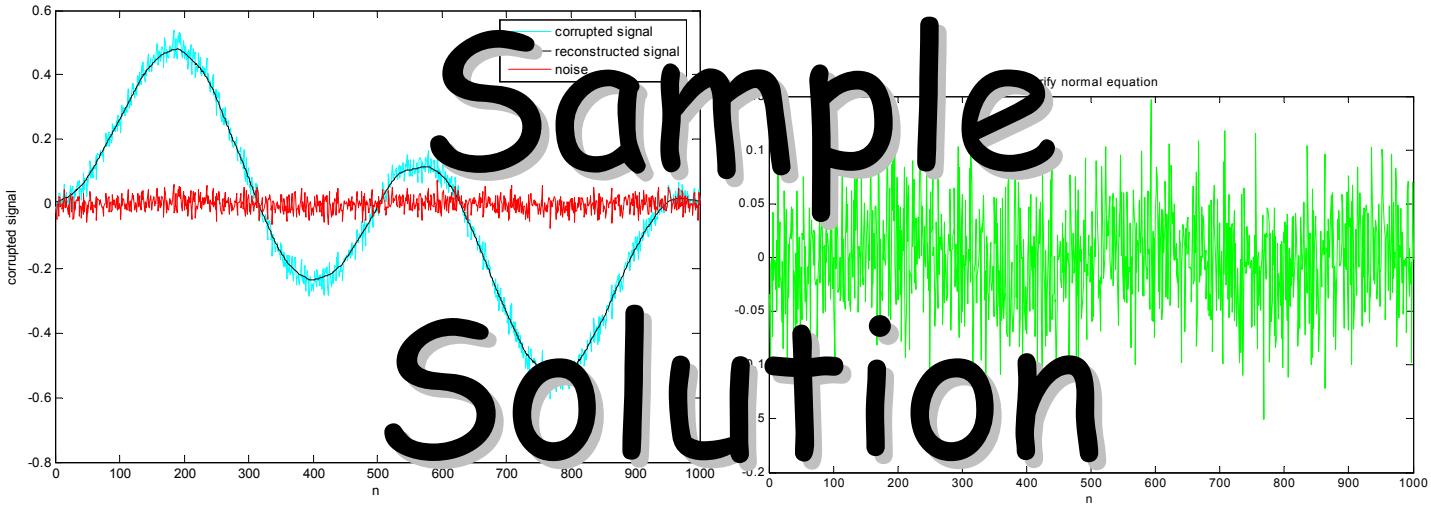
$$(I + \lambda D^T D)x = x_{cor} \text{ .....(3)}$$

MATLAB® provides special routines for solving sparse linear equations, and they are used as follows. There are two types of matrices: full (or dense) and sparse. If you define a matrix, it is considered full by default, unless you specify that it is sparse. You can convert a full matrix to sparse format using the command A = sparse(A), and a sparse matrix to full format using the command A = full(A). When you type x = A \ b where A is $n \times n$, MATLAB® chooses different algorithms depending on the type of A. If A is full it uses the standard methods for general matrices. If A is sparse, it uses an algorithm that takes advantage of sparsity. In our application, the matrix $I + \lambda D^T D$ is sparse (in fact tridiagonal), so if we make sure to define it as a sparse matrix, the normal equations will be solved much more quickly than if we ignore the sparsity. The command to create a sparse zero matrix of dimension $m \times n$ is A = sparse(m,n). The command A = speye(n) creates a sparse $n \times n$-identity matrix. If you add or multiply sparse matrices, the result is automatically considered sparse. This means you can solve the normal equations (3) by the following MATLAB® code (assuming λ and x_{cor} are defined):

```
D = sparse(999,1000);
D(:,1:999) = -speye(999);
D(:,2:1000) = D(:,2:1000) + speye(999);
xhat = (speye(1000) + lambda*D'*D) \ xcor;
```

Solve the least squares problem (1) with the vector x_{cor} defined in denoise.m, for various values of λ: 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, and $\lambda = 10^5$. Among them, plot the four reconstructed signals $\hat{x}$ at least. Discuss the effect of λ on the quality of the estimate $\hat{x}$. Compress and submit your report including matlab® codes by email to teacher and submit a copy of printed report during the class on May 16 simultaneously.

Example: For a certain λ, as shown in the left plot, we can estimate the original signal from the corrupted signal. Also, we can verify the nominal equation in (3), as shown in the right plot.



Instruction: You will receive 80 pts, if you work out the signal reconstruction by the process above. You have to test and discuss your codes and your results. This is the basic requirement on this homework. Only plots without sufficient discussion, you can get merely 70 pts. You will get additional 20 pts if you can extend this problem to deal with more complicated issues of the denoising, such as multi-noise appearing in the signal. This part is open and has no limitation.