Numerical Analysis - Homework #4

Due Data : May 08, 2020 (23:59pm)

On-line Submission Only (Report and Source Code)

Problem 1 [70pts]. Consider the function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ defined by

$$f(\boldsymbol{x}) = \frac{1}{2}(x_1^3 - x_2)^2 + \frac{1}{2}(x_1 - 1)^2$$

(a) At what point does f attain a minimum?

(b) Perform one iteration of Newton's method for minimizing f using as a starting point $\boldsymbol{x}_0 = [2 \ \ 2]^T$.

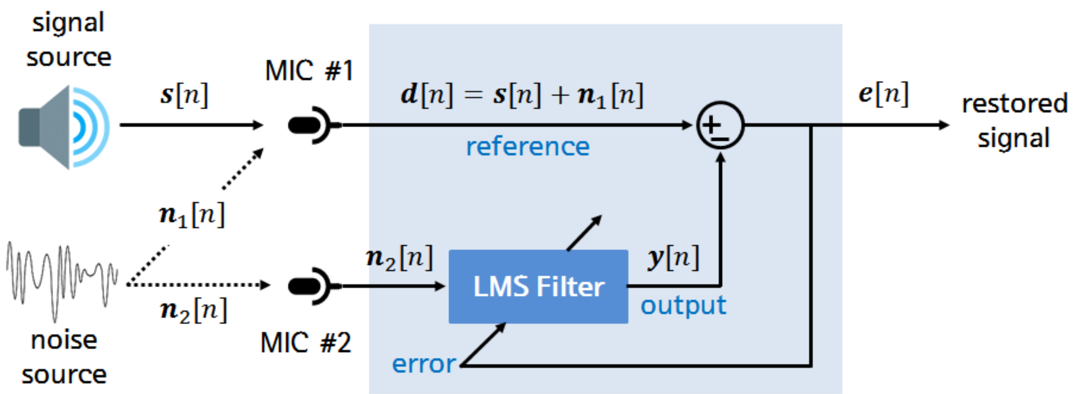
(c) Is $\boldsymbol{x}_1$ closer than $\boldsymbol{x}_0$ to the minimum?

(d) Is $f(\boldsymbol{x}_1)$ less than $f(\boldsymbol{x}_0)$?

Problem 2 [15pts]. Using your own C-program, find the minimum point of the following function using the gradient descent and steepest descent algorithms with starting location $\boldsymbol{x}_0 = [-2 \quad 1]^T$. Compare their performances in view of convergence rate(speed).

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{b}, \ \ A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \ \ \boldsymbol{b} = \begin{bmatrix} 4 \\ -2 \end{bmatrix}$$

Problem 3 [15pts]. Consider the block diagram of an adaptive noise canceller. It is interesting that the similar concept can be found in your wireless earphone.



adaptive noise cancellation

In the above, the ultimate goal of the noise cancellation is to recover the signal source of interest from the available data measured by MIC #1 and MIC #2, respectively. Note that, apart from the MIC #1, the additional MIC #2 records the noise signal $n_2[n]$ only. Since there are two different paths from the noise source to the microphones, it is obvious that $n_1[n] \neq n_2[n]$, but the frequency of $n_1[n]$ is the same with that of $n_2[n]$. Exploiting this fact, if the recorded noise $n_2[n]$ is filtered to make it more like $n_1[n]$, we can recover the original signal $s[n]$.

The following signal models are often be used for designing the noise canceller.

signal source	(reference)	$d[n]$
noise source	(input)	$n_2[n]$
filter taps		$\boldsymbol{h} = [h_0 \quad h_1 \quad \cdots \quad h_{M-1}]^T$
filter output		$y[n] = \sum_{k=0}^{M-1} h_k n_2[n-k] = \boldsymbol{u}^T[n]\boldsymbol{h}$, $\boldsymbol{u} = [n_2[n] \quad n_2[n-1] \quad \cdots \quad n_2[n-M+1]]^{\boldsymbol{T}}$
restored signal	(error)	$e[n] = d[n] - y[n]$

Using the above signal models, the design problem of the adaptive noise canceller reduces the problem of finding the filter taps $\boldsymbol{h}$ minimizing the cost function defined as

$$J(\boldsymbol{h}) = E\{e^2[n]\} = E\{d^2[n]\} + \boldsymbol{h}^T E\{\boldsymbol{u}[n]\boldsymbol{u}^T[n]\}\boldsymbol{h} - 2\boldsymbol{h}^T E\{\boldsymbol{u}[n]d[n]\}$$

Therefore, the adaptation in the negative direction of the gradient becomes

$$\boldsymbol{h}[n+1] = \boldsymbol{h}[n] - \frac{\mu}{2} \cdot \nabla J(\boldsymbol{h}[n])$$

where

$$\nabla J(\boldsymbol{h}[n]) = \left.\frac{\partial J(\boldsymbol{h})}{\partial \boldsymbol{h}}\right|_{\boldsymbol{h}=\boldsymbol{h}[n]} = 2E\{\boldsymbol{u}[n]\boldsymbol{u}^T[n]\} - 2E\{\boldsymbol{u}[n]d[n]\} = 2E\{\boldsymbol{u}[n]e[n]\}$$

Unfortunately, the expectation of $\boldsymbol{u}[n]e[n]$ is often unknown in practice, hence remedy is to replace it with the instantaneous approximation. As a result, the steepest descent algorithm can be rewritten as

$$e[n] = d[n] - \boldsymbol{h}^T[n]\boldsymbol{u}[n]$$

$$\boldsymbol{h}[n+1] = \boldsymbol{h}[n] - \mu \cdot \boldsymbol{u}[n]e[n]$$

It is well-known that the least mean squares(LMS) estimator is equal to the steepest descent algorithm.

(a) In your C-program, load the data files, “mic1.dat” and “mic2.dat” which contain $d[n]$ and $n_2[n]$ respectively.

(b) For $M = 2$, write down your own C program of the LMS estimator and find the restored signal $e[n]$.

Plot the following figures and discuss your results using them.

Figure 1: signals $d[n]$ and $e[n]$

Figure 2: filter taps $h_0[n]$ and $h_1[n]$

Figure 3: signals $n_2[n]$ and $y[n]$

c.f.)

It is helpful to listen the sounds $d[n]$, $n_2[n]$ and $e[n]$ using Matlab command, “**sound**”.

e.g. sound(data, 44100)