Boston University
Department of Electrical and Computer Engineering

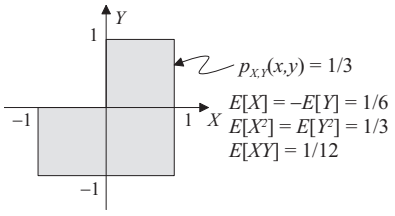
EC505 STOCHASTIC PROCESSES

Problem Set No. 9

Fall 2016

Issued: Monday, Nov. 28, 2016 **Due:** Wednesday, Dec. 7, 2016

The random variables X and Y are uniformly distributed over the region shown in the figure.



(a) Find $\widehat{x}_{BLS}(y)$ the Bayes least square estimate of X given Y and λ_{BLS}, the associated error variance.

(b) Find $\widehat{x}_{LLS}(y)$ the linear least square estimate of X based on Y and λ_{LLS} the associated error variance.

(c) Consider the following "modified least squares" cost function:

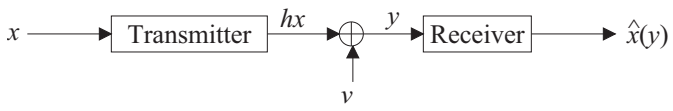
$$J(\widehat{x}, x) = \begin{cases} K(\widehat{x} - x)^2 & x > 0 \\ (\widehat{x} - x)^2 & x < 0 \end{cases}$$

where $K > 1$ is a constant. Determine $\widehat{x}_{MLS}(y)$, Bayes estimate of X corresponding to this cost criterion.

Hint: Examine the conditional form of the defining Bayes optimization problem and see if you can find a similar problem with the standard structure but a different density.

Problem 9.1

Consider the communication system shown below. The message X is an $N(0, \sigma_x^2)$ random variable. The transmitter output is HX, and the receiver input is: $Y = HX + V$, where V is an $N(0, r)$ random variable that is statistically independent of X.



Suppose the transmitter is subject to intermittent failure, i.e., H is a random variable taking the values 0 and 1 with probabilities $1-p$ and p, respectively. Assume H is statistically independent of both X and V.

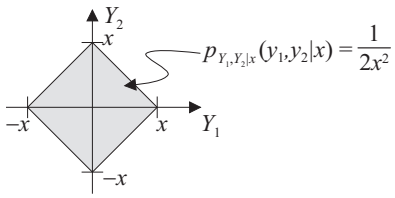
(a) Find $\hat{x}_{LLS}(y)$, the linear least-squares estimate of X based on observation of Y, and λ_{LLS}, its resulting error variance and mean-square estimation error.

(b) Show that: $E[X \mid Y = y] = \sum_{i=0}^{1} \Pr[H = i | Y = y]\, E[X \mid Y = y, H = i]$. Hint: Consider using iterated expectation.

(c) Find $\hat{x}_{BLS}(y)$, the Bayes least-squares estimate of X based on observation of Y.

Problem 9.2 (Old Exam Problem)

We wish to estimate an unknown, deterministic parameter x from two observations y_1 and y_2 that are uniformly distributed over the diamond shaped region parameterized by $x > 0$ shown in the figure. Hint: All parts can be answered with minimal calculation with some thought.



(a) Find $p_{Y_1|x}(y_1|x)$.

(b) Find a maximum-likelihood estimate of x based on y_1 alone.

(c) Find a maximum-likelihood estimate of x given both y_1 and y_2.

Problem 9.3

The purpose of this problem is to determine the (unknown) probability of heads when flipping a particular coin.

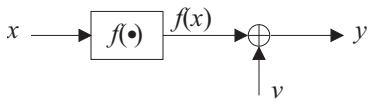
(a) Suppose that the coin is flipped N times in succession, each toss statistically independent of all others, each with (unknown) probability p of heads. Let n be the number of heads that is observed. Find the maximum likelihood (ML) estimate $\widehat{p}_{ML}(n)$ of p based on knowledge of n.

(b) Evaluate the bias $E[p - \widehat{p}_{ML}(n) \mid p]$ and the mean-square error $E[(p - \widehat{p}_{ML}(n))^2 \mid p]$ of the ML estimate.

(c) Is the ML estimate efficient (i.e. does the mean square error attain the Cramer-Rao bound)? An estimate is termed "consistent" if its mean square error goes to zero as $N \to \infty$. Is the ML estimate of p consistent. Briefly explain.

Problem 9.4

A hypothetical communication system is shown the figure below:



The deterministic parameter x is unknown, and the receiver noise V is assumed to be a Gaussian random variable with mean 0 and variance σ^2. The function $f(\cdot)$ is given by

$$f(x) = (1 - e^{|x|}) \operatorname{sgn}(x)$$

where $\operatorname{sgn}(x)$ denotes the sign of x.

(a) Find the maximum likelihood estimate of x based on observation of Y.

(b) Find an explicit lower bound on the mean-square estimation error of an arbitrary unbiased estimate of x based on observation of Y.

Problem 9.5

Suppose we observe a random N-dimensional vector $\underline{Y}$, whose components are independent, identically-distributed Gaussian random variables, each with mean x_1 and variance x_2.

(a) Suppose the mean x_1 is unknown but the variance x_2 is known. Find the maximum likelihood estimate of the mean x_1 based on observation of $\underline{Y}$. Evaluate the bias and the mean-square error for this estimate. Is this estimate efficient?

(b) Suppose the mean x_1 is known but the variance x_2 is unknown. Find the maximum likelihood estimate of the variance x_2 based on observation of $\underline{Y}$. Evaluate the bias and the mean-square error for this estimate. Is this estimate efficient? Explain. Note: If $z \sim N(m, \lambda)$, then $(z-m)^2/\lambda$ is a Chi-squared random variable with one degree of freedom, which has variance 2.

(c) Suppose both the mean x_1 and the variance x_2 are unknown. Find $\underline{\widehat{x}}_{ML}$, the maximum likelihood estimate of:

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

based on observation of $\underline{Y}$. Evaluate the bias in both component estimates and $E[(x_1 - \widehat{x}_{1_{ML}})^2]$. Evaluate $E[(x_2 - \widehat{x}_{2_{ML}})^2]$ for $N = 2$. Compare with your results from parts (a) and (b). Note: If $z \sim N(0, \lambda)$, then $E[z^n] = \lambda^{n/2} \cdot 1 \cdot 3 \cdot 5 \cdots (n-1)$ if n is even and $E[z^n] = 0$ if n is odd.

Problem 9.6

Suppose we observe the process:

$$y(t) = x(t) + v(t)$$

where $x(t)$ and $v(t)$ are uncorrelated, zero-mean processes, with

$$S_{XX}(s) = \frac{3}{1-s^2} \qquad S_{VV}(s) = \frac{5}{9-s^2}$$

Find the noncausal Wiener filter for estimating $x(t)$ based on $y(t)$. Also find the corresponding mean-square estimation error.

Problem 9.7

Let $x(t)$ be a zero-mean wide-sense stationary stochastic process with covariance function $K_{XX}(t)$, and suppose we observe $x(0)$ and $x(T)$.

(a) What is the linear least-squares estimate of $x(t)$ for any $0 < t < T$, based on $x(0)$ and $x(T)$? Hint: Given its a linear estimate, start with the form of the estimator and then use orthogonality conditions to obtain a set of equations for the coefficients.

(b) What is the resulting mean-square estimation error?

(c) Evaluate parts (a) and (b) for $t = T/2$.

(d) Let $d = \int_0^T x(u)\,du$ Find the linear least-squares estimate of d based on $x(0)$ and $x(T)$, and compute the resulting mean-square estimation error. Hint: Again follow the procedure described in part (a).

Problem 9.8

Let $x(n)$ be a signal that you want to estimate from noisy observations $y(n) = x(n) + v(n)$. The noise is white ($S_{VV}(z) = 1$) and independent of $x(n)$ and the signal is a first-order auto-regressive process with power spectral density in the $\mathcal{Z}$ domain:

$$S_{XX}(z) = \frac{1}{(1-0.9z)(1-0.9z^{-1})}$$

Find the causal Wiener filter for estimating $x(n)$, and its associated mean-squared error. Note that the causal Wiener filter is the same as the discrete-time Kalman filter. You will have to first represent the system in state-space form. Then compute the Kalman Filter for the resulting state-space system.

Problem 9.9

(a) (Shanmugan and Breipohl 7.30) Suppose the discrete-time random process $x(t)$ obeys the following AR difference equation:

$$\begin{aligned} x(t+1) &= 0.9x(t) + w(t) \\ y(t) &= x(t) + v(t) \\ E[w(t)w(s)] &= 1\delta(t-s) \\ E[v(t)v(s)] &= 1\delta(t-s) \\ m_x(1) = 0, &\quad P_x(1) = 10 \end{aligned}$$

where $v(t)$ and $w(t)$ are independent of each other. The following data is observed:

$$y(1) = 1, \quad y(2) = 1.1, \quad y(3) = 1.2, \quad y(4) = .9, \quad y(5) = 1.2$$

Use the Kalman filtering formulas to find $\widehat{x}(t|t-1)$, $P(t|t-1)$, $\widehat{x}(t|t)$, $P(t|t)$, and the Kalman gain $K(t)$ for $t = 1, \ldots, 6$. Use Matlab or a spreadsheet to do the calculation and make a table like that below to keep your answers organized.

t	$\widehat{x}(t\|t-1)$	$y(t)$	$P(t\|t-1)$	$K(t)$	$\widehat{x}(t\|t)$	$P(t\|t)$	$\widehat{x}(t+1\|t)$	$P(t+1\|t)$
1								
2								
3								
4								
5								
6								

(b) (Shanmugan and Breipohl 7.31) What is the steady state Kalman gain $\lim_{t\to\infty} K(t)$ for part (a)? What is the steady state error covariance $\lim_{t\to\infty} P(t|t) = P$.

(c) Consider the input/output relationship between $\widehat{x}(n|n)$ and $y(n)$ corresponding to your Kalman filter solution in the steady state and show that the Kalman filter is the same as the causal Wiener filter by finding the Kalman filter system function $H_k(z)$. Look up the notes and verify that the solution you get here is the same as what you would get if you solved for a causal wiener filter.

Computer Problems

Problem 9.10 2-D MAP Estimation Based on Brownian Motion Priors

In this problem we extend the results of the 1-D Problem 9.6 to 2-D through the use of a 2-D Brownian motion-type prior model to estimate an image from noisy observations. This 2-D Brownian motion prior model will be a simple example of a *Markov Random Field* model. Note that we will be processing large arrays of numbers in MATLAB in this project, so you will need a computer with a reasonable amount of RAM or you will run into trouble. In addition, we will need to be careful about creating the large covariance matrices or their inverses needed to find the MAP estimate or we will run out of memory. We will need to use MATLAB's sparse matrix capabilities (see `help sparfun` in MATLAB).

2-D Prior Model: To proceed, we need to define what we mean by a 2-D Brownian motion prior model. A natural approach, and the one we will take, is to develop a model of the form of equation (7) of Problem Set 9. Our first issue is to define what we mean by the vector $\underline{x}$. For our 1-D problems, this vector was naturally defined to be the ordered time points of the process. For an image there is no similar, obvious ordering of the points, so we will simply choose a convention and use it throughout this problem. The most

common mapping of an $N_1 \times N_2$ array of image pixels x_{n_1,n_2} to an $N = N_1 N_2$ vector $\underline{x}$, and the one we will use, is obtained by *stacking* one column on top of another, as follows:

$$[x_{n_1,n_2}] = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N_2} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N_2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N_1,1} & x_{N_1,2} & \cdots & x_{N_1,N_2} \end{bmatrix} \quad \Longrightarrow \quad \underline{x} = \left[\begin{array}{c} x_{1,1} \\ \vdots \\ x_{N_1,1} \\ \hline x_{1,2} \\ \vdots \\ x_{N_1,2} \\ \hline \vdots \end{array}\right] \tag{1}$$

In MATLAB this operation taking a 2-D image `X` (i.e. a matrix) to its vector representation `x` is performed using the colon operator ":", so that `x = X(:);`. The opposite operation taking a vector representation back to an $N_1 \times N_2$ image matrix is done using the MATLAB function `reshape.m`, as follows: `X = reshape(x,N1,N2);`.

Our next issue is how to actually define an appropriate 2-D Brownian motion prior model that we can use for estimation. Using equation (7) of Problem Set 9 as motivation, we will define our 2-D Brownian motion prior model explicitly as:

$$\underline{x} \quad \sim \quad N\left(0, q\left(D_t^T D_t\right)^{-1}\right) \tag{2}$$

where q represents the overall prior variance and we define the "total" derivative operator D_t for the 2-D case as:

$$D_t \quad \equiv \quad \left[\frac{D_1}{D_2}\right] \tag{3}$$

where the $N \times N$ matrices D_1 and D_2 are discrete first partial derivative operators along the n_1 (i.e. the "x") and n_2 (i.e. the "y") directions, respectively, so that the total derivative operator D_t is formed by *stacking* D_1 over D_2. Thus, the vector $D_1\underline{x}$ is the image gradient in the n_1 direction while the vector $D_2\underline{x}$ is the image gradient in the n_2 direction. This prior model extends the 1-D prior model by simply using the 2-D gradient in place of the 1-D derivative.

The model (2) gives a prior covariance defined by $Q = q\left(D_t^T D_t\right)^{-1}$. We will use this model as the prior in our estimate. Given the above definitions we can see that:

$$\underline{x} \quad \sim \quad N\left(0, q\left(D_1^T D_1 + D_2^T D_2\right)^{-1}\right) \tag{4}$$

While it is not necessary for our purposes, note that an associated *implicit* prior model for $\underline{x}$ which is consistent with (2) can be defined by:

$$D_t^T D_t \underline{x} \quad = \quad \underline{v}', \qquad \underline{v}' \sim N\left(0, q D_t^T D_t\right) \tag{5}$$

This implicit model is fundamentally different than in the 1-D case. In particular, note that a *correlated* driving noise $\underline{v}'$ is now needed. Further, the covariance of this noise is the *inverse* of the prior covariance and the implicit model matrix on the left hand side is also essentially the inverse of the prior covariance. In the 1-D case the driving noise was white and the model matrix on the left hand side was the square root of the inverse of the prior covariance. These differences in the 1-D and 2-D models reflect the differences between 1-D and 2-D AR models. Basically, while causality of an AR model in 1-D can always be obtained through appropriate redefinition of the time axis, in 2-D this may not be possible. It is the constraints imposed by our inability to find a total ordering of the points in 2-D that require a correlated driving noise.

One way of understanding what is happening is to consider that in 2-D there are many paths connecting any two pixels. To relate the values between these two pixels we may add up the increments along any such

path. Since the paths must end up in the same place, the increments cannot be independent. In short, life is much more complicated in 2-D! A detailed treatment of such issues is beyond our present scope.

Observation Model: Lastly, we will need an observation model. This observation model will be given by:

$$y_{n_1,n_2} = x_{n_1,n_2} + w_{n_1,n_2}, \quad w_{n_1,n_2} \sim N(0, r_{n_1,n_2}) \tag{6}$$

For this problem we will again use a stationary observation noise model with $r_{n_1,n_2} = 400$. In terms of the vector representation of the image fields this observation model is given by:

$$\underline{y} = \underline{x} + \underline{w}, \quad \underline{w} \sim N(0, R) \tag{7}$$

where $R = 400I$.

(a) We will use the "trees" image in MATLAB as our truth image `X` and create data by corrupting this image with additive Gaussian noise according to (8) or (7). To this end, load the trees image `X` (e.g. `load trees`), create the associated noisy observed image `Y` as described in (8) and then find the corresponding vector representations `x` and `y`. Also define the corresponding noise covariance matrix `R` of the observation process for later use in estimation. You will want to create `R` as a sparse matrix using the MATLAB function `speye.m` (which only saves the nonzero entries) to save space.

Display the original image `X` and the noisy data image `Y` together using the MATLAB function `imagesc.m` . Make sure you use the same grayscale colormap for all your images so you can make a fair comparison. You can do this in MATLAB as follows (which we demonstrate for `X`):

```
imagesc(X,[1 128]);
axis image
colormap gray
```

The second argument to `imagesc.m` sets the intensity limits of the image. What is the mean square error of the noisy data? Save your data `X` and `Y` for use in Problem Set 11.

(b) To perform MAP estimation of `x` based on `y` we need to define a prior. We will use the model in (2), but need to find the discrete partial derivative matrices D_1 and D_2 as well as a reasonable value for the overall driving noise power q. On the web site you will find a MATLAB function `dgrad.m` which will produce the matrices `D1` and `D2`. To choose a value for q we will use the interpretation of q as the variance of the increments or jumps in the noise free process per discrete coordinate step. This suggests a reasonable choice for q should be related to the variance of the derivative of the noise free image $D_t\underline{x}$. We will (somewhat arbitrarily) set $q = 3 * \mathrm{Var}\left[D_t\underline{x}\right]$

Use the supplied function `dgrad.m` to find the total derivative matrix `Dt` and set `q` as described above. The function `dgrad.m` creates the derivative operators `D1` and `D2` as sparse matrices to save space. Note that in this 2-D case we *will not* find the explicit prior process covariance Q, as the computation and memory requirements necessary to find and store Q are prohibitive. To find the MAP estimate we do not need Q, but rather only its *inverse*, which may be written directly in terms of `Dt`.

(c) Use the problem elements found in parts (a) and (b) to find the vector representing the MAP estimate `xmap` of the image `x` from the noisy data in `y`. Find the corresponding image estimate `Xmap` by reshaping `xmap`. This can be done in MATLAB as follows:

```
xmap = ( inv(R) + inv(q)*Dt'*Dt )\(inv(R)*y);
Xmap = reshape(xmap,N1,N2);
```

Make sure you are representing `R` and `Dt` as sparse matrices, or you will run out of memory! Also, do not try to explicitly invert the MAP normal equation, but rather make use of MATLAB's "backslash"

operator as shown above. This computation will take a significant amount of time even on a fast computer (e.g. 90 seconds on an Ultrasparc).

Display the original image `X`, the noisy data image `Y`, and the MAP estimate image `Xmap` together. Again, be careful to use the same grayscale colormap as discussed in (a). What is the mean square error of the MAP estimate? How does the MAP estimate compare with the original image and the noisy data in terms of MSE and appearence? What are some advantages to this approach to denoising images? What are some problems with this approach? Later, we will compare this "time domain" approach with the frequency domain approach corresponding to the Wiener filter.

Problem 9.11 The Two-Dimensional Discrete-Space Wiener Filter

In this problem we will extend the Wiener filter theory to 2-D and implement a discrete-space 2-D noncausal Wiener filter. We will apply this filter to the restoration of the same noisy image we have examined and compare the results. As you know, the Wiener filter is the linear minimum square estimator of a stationary process, and thus for stationary Gaussian processes can be viewed as a frequency domain implementation of the MAP estimator we have examined in these earlier problem sets.

(a) As we did in Problem Set 10, we will use the "trees" image in MATLAB as our truth image `X` and create data by corrupting this image with additive Gaussian noise according to the model:

$$y_{n_1,n_2} = x_{n_1,n_2} + w_{n_1,n_2}, \quad w_{n_1,n_2} \sim N(0, r_{n_1,n_2}) \tag{8}$$

For this problem we will again use a stationary observation noise model with $r_{n_1,n_2} = 400$. Either use the same noisy image you created in Problem Set 10 or load the trees image `X` and recreate the associated noisy observed image `Y` as described in (8). What is the mean square error of the noisy data? In this problem we will not need to represent the images by vectors as we did in Problem Set 10.

You may wish to display the original image `X` and noisy data image `Y` as described in Problem Set 10. Remember to use a common grayscale colormap.

(b) From class we know that the 1-D noncausal Wiener filter $H(f)$ of the 1-D process x_n based on observation of the 1-D process $y_n = x_n + w_n$, where w_n is white noise is given by:

$$H(f) = \frac{S_{YX}(f)}{S_{YY}(f)} = \frac{S_{XX}(f)}{S_{XX}(f) + S_{WW}(f)} \tag{9}$$

For the 2-D case given in (8), the expression for the Wiener filter is exactly the same, only 2-D spectral densities are used to define an equivalent 2-D filter:

$$H(f_1, f_2) = \frac{S_{YX}(f_1, f_2)}{S_{YY}(f_1, f_2)} = \frac{S_{XX}(f_1, f_2)}{S_{XX}(f_1, f_2) + S_{WW}(f_1, f_2)} = \frac{S_{XX}(f_1, f_2)}{S_{XX}(f_1, f_2) + 400} \tag{10}$$

where we have used the fact that $S_{WW}(f) = 400$ for the problem defined in (8).

Thus to apply the Wiener filter we only need to find the 2-D power spectral density $S_{XX}(f_1, f_2)$ of the image X. We will use a simple estimate of $S_{XX}(f_1, f_2)$ based on our previous work on spectral estimation. In particular, we will use the "periodogram" estimate of the power spectral density based on the noise-free image X. Recall that the periodogram estimate of $S_{XX}(f_1, f_2)$ is nothing more than the magnitude squared of the 2-D Fourier transform of X scaled by the number of points in the field: $S_{XX}(f_1, f_2) = |\mathcal{F}[X(n_1, n_2)]|^2/N$, where $\mathcal{F}[\cdot]$ represents the 2-D Fourier transform. Estimate $S_{XX}(f_1, f_2)$ based on this formula in MATLAB as follows:

```
Sxx = (abs(fft2(X)).^2)/N;
```

The PSD you have estimated provides an estimate of the correlation function $R_{XX}(n_1, n_2)$ of the process. Does the estimated $R_{XX}(n_1, n_2)$ that you have computed seem reasonable?

(c) Use the $S_{XX}(f_1, f_2)$ you found in part (b) together with the formulas in (10) to calculate the frequency response $H(f_1, f_2)$ of the 2-D non-causal Wiener filter for this problem. You can do a point-by-point division in MATLAB using the dot-divide operator "./". Display the magnitude of this frequency response as an image using `imagesc.m`. You may wish to use the matlab function fftshift.m to shift the zero frequency part of the spectrum to the center of the image. What type of filter is this (high pass, low pass, or band pass)?

(d) Apply the Wiener filter `H` you found in part (c) to the noisy data in `Y` to obtain the corresponding Wiener filter estimated signal `Xwf`. This simply involves a point by point product of the two signals in the frequency domain followed by 2-D inverse Fourier transformation. In MATLAB this is done as follows:

```
Xwf = real(ifft2(fft2(Y).*H));
```

where we take the real part of the solution because of numerical round-off. What is the mean square error of the estimate `Xwf`?

Using a common gray scale colormap, display the true image `X`, the noisy signal `Y`, and the Wiener filtered image `Xwf`. Compare the Wiener filtered estimate to the MAP estimate you found in Problem Set 10. Which approach is more computationally efficient? The Wiener filtered or MAP estimated images certainly appear to have reduced noise and lower mean square error. Are there any problems with these estimated images? What degradation exists in these estimated images relative to the original images?