Problem 1. Random variables as vectors.
Let $X, Y, Z$ be zero-mean, unit-variance, real-valued random variables which satisfy
\[ \text{Var}(X + Y + Z) = 0. \]

Find the covariance matrix of $X, Y, Z$, i.e., find the matrix
\[
\begin{pmatrix}
E[X^2] & E[XY] & E[XZ] \\
E[XZ] & E[YZ] & E[Z^2]
\end{pmatrix}.
\]

Problem 2. A Kalman filter and another recursive linear estimator.
Suppose we would like to estimate $S$ from its noisy observations:
\[ Y(n) = S + W(n), \quad n = 0, 1, \ldots \]  

Our model for $S$ is that it is a zero-mean, unit-variance random variable. $W(n)$ is a zero-mean, unit-variance white noise process, uncorrelated with $S$.

(a) Derive a recursive formula for $\hat{S}_n$, the linear least-squares estimate of $S$ based on observing $Y(0), Y(1), \ldots, Y(n)$. Derive a recursive formula for $\lambda_n$, the corresponding error variance. (Hint. Argue that Eq. (1) is equivalent to the following:
\[
\begin{align*}
S(0) &= S \\
S(n) &= S(n-1) + X(n), \quad n \geq 1 \\
Y(n) &= S(n) + W(n), \quad n = 0, 1, \ldots,
\end{align*}
\]
where the variance of the white, zero-mean noise $X(n)$ is zero. Now apply the Kalman filtering formulas derived in class.)

(b) Prove that $\lambda_n = \frac{1}{n+2}$. (Assume $\lambda_{-1} = 1$.)

(c) From http://www.ece.purdue.edu/~ipollak/ee438/FALL03/homeworks/hw.html download the file y.mat. This file contains three 50-point discrete-time sequences, called $y_1, y_2, y_3$. Download them into the Matlab workspace, using the command load y. Apply the filter derived in Part (a) to the sequence $y_1$. Turn in your Matlab code and the plots of $y_1, \hat{S}_n$ (as a function of $n$) and $\lambda_n$. Using the Matlab command mean(y1), calculate the average of the signal $y_1$. Note that $\hat{S}_{50}$ is the linear least-squares estimate of $S$ based on all the 50 observations. Is it equal to the average of the signal $y_1$? Explain why or why not.

(d) Consider the following alternative recursive estimator of $S$:
\[
\begin{align*}
\hat{S}'_{-1} &= 0; \\
\hat{S}'_n &= \hat{S}'_{n-1} + 0.5(Y(n) - \hat{S}'_{n-1}), \quad n = 0, 1, \ldots
\end{align*}
\]
Implement it in Matlab, and apply it to $y_1$. Turn in your plot of $\hat{S}'_n$. 

\[ \text{ECE 438} \]
\[ \text{Homework 12, due in class Wednesday, 12/10/2003.} \]
(e) Apply the two estimators (from Part (a) and from Part (d)) to the signal $y_2$, and turn in the corresponding plots of $\hat{S}_n$ and $\hat{S}'_n$, and a plot of $y_2$. To see the behavior of the two estimators more clearly, you might want to plot $\hat{S}_n$ and $y_2$ on the same plot (in two different colors), and then $\hat{S}'_n$ and $y_2$ on the second plot.

(f) Apply the two estimators (from Part (a) and from Part (d)) to the signal $y_3$, and turn in the corresponding plots of $\hat{S}_n$ and $\hat{S}'_n$, and a plot of $y_3$. Similarly to Part (e), it may be better to place $\hat{S}_n$ and $y_3$ on the same plot, and then $\hat{S}'_n$ and $y_3$ on the same plot.

(g) Discuss the advantages and shortcomings of the two estimators. Discuss settings in which one or the other may be more appropriate.

**Problem 3.** In the above picture, all the images are $20 \times 20$ discrete-space, continuous-valued, grayscale images. Each pixel can have values from 0 (black) to 3 (white). The image in the top row depicts a light square on a dark background. Within the square, the image has a constant value of 2; within the background, it has a constant value of 1. (The image may look somewhat grainy and non-constant on a printout because of imperfections of printing and copying devices.) Each of the six images in the second and third rows was obtained by modifying the square image in the top row. The objective of this problem is to determine what operations resulted in the six images.

a. One of the images in the second row was obtained from the square image in the top row by adding white zero-mean noise with marginal density which is uniform between -1 and 1. One of the images
in the second row was obtained from the square image in the top row by adding white zero-mean Gaussian noise with standard deviation 0.2. One of the images in the second row was obtained by corrupting the square image in the top row with “occlusive” noise: each pixel value stayed the same with probability 0.9, was replaced by 0 with probability 0.05, and was replaced by 3 with probability 0.05. Indicate which image in the second row (left, center, right) corresponds to each type of noise (uniform, Gaussian, occlusive), and fully explain your answers.

b. The three images in the third row were obtained by applying a median filter with a $3 \times 3$ window to the three images in the second row. Indicate which image in the third row corresponds to each image in the second row, and fully explain your answers.

Problem 4. In Matlab, implement the iterative non-linear filter (the Perona-Malik filter) considered in class:

$$
g^{(i+1)}(m, n) = g^{(i)}(m, n) + \alpha \left\{ F[g^{(i)}(m + 1, n) - g^{(i)}(m, n)] + F[g^{(i)}(m - 1, n) - g^{(i)}(m, n)] + F[g^{(i)}(m, n + 1) - g^{(i)}(m, n)] + F[g^{(i)}(m, n - 1) - g^{(i)}(m, n)] \right\}, \quad (2)
$$

where $M$ and $N$ are the dimensions of the image, $I$ is the total number of iterations, and $F(v)$ is the following nonlinear function which depends on the positive real parameter $K$:

$$F(v) = \frac{v}{1 + (\frac{v}{K})^2}, \text{ for any real number } v.$$

Use the following syntax:

```
function g = PM(f, alpha, I, K)
```

Here, $f = g^{(0)}$ is the input image to be filtered; alpha is the filter parameter $\alpha$; $I$ is the total number of iterations; $K$ is the filter parameter $K$; and $g = g^{(I)}$ is the output image, which is the result of $I$ iterations of the filter.

To calculate $g^{(i+1)}(m, n)$ around the border of the image, assume that $g^{(i)}(m, n)$ is symmetrically reflected. In other words, assume that

$$
\begin{align*}
g^{(i)}(0, n) & = g^{(i)}(1, n) \\
g^{(i)}(m, 0) & = g^{(i)}(m, 1) \\
g^{(i)}(M + 1, n) & = g^{(i)}(M, n) \\
g^{(i)}(m, N + 1) & = g^{(i)}(m, N)
\end{align*}
$$

Hints. Avoid “for” loops. Your program should have only one “for” loop: to iterate Eq. (2) over $i = 0, 1, \ldots I - 1$. All other computations should be done on entire images. For example, to calculate $F[g^{(i)}(m + 1, n) - g^{(i)}(m, n)]$, you can use

```
delta_g = [g(2:M,:), g(M,:)] - g;
F_of_delta_g = delta_g ./ (1 + (delta_g / K).^2);
```

(a) Test your program on a noisy binary image, created with the following script:
rand('state',sum(100*clock));
test_im = [ones(50,25)*50 ones(50,25)*200]+floor(rand(50,50)*50);

Use alpha= 0.25, K = 2, and 200 iterations:

\[ g = PM(test\_im, 0.25, 200, 2); \]

Display both the noisy image \textit{test\_im} and the filtered image \textit{g}. Recall from Lab 10 that the following sequence of commands should be used to display a grayscale image:

\[ \text{image(test\_im);} \]
\[ \text{colormap(gray(256));} \]
\[ \text{axis('image');} \]

(b) Use your program to filter the noisy race-car image noise1.tif from Lab 10. Use the following four different parameter settings.

(i) Use alpha= 0.25, K = 2, and 200 iterations:

\[ g = PM(\text{double(noise1)}, 0.25, 200, 2); \]
(This could take 1-2 minutes.)

(ii) alpha= 0.25, K = 2, 500 iterations.

(iii) alpha= 0.25, K = 20, 20 iterations.

(iv) alpha= 0.25, K = 20, 50 iterations.

Display the original noise-free image \textit{race}, the noisy image \textit{noise1} and the four filtered images. Use \texttt{subplot} to put the images on the same page. Comment on the quality of the reconstruction in each of the four cases, and explain the differences among the four results.

(c) Another nonlinear method of noise removal discussed in class is to decompose an image in a basis, set to zero all coefficients that are smaller than some threshold, and then reconstruct an estimate from the remaining coefficients. This procedure, for a wavelet basis, is performed by the Matlab command \texttt{wdencmp} which is a part of the Wavelet Toolbox. The syntax is:

\[ \text{out = wdencmp('gbl',noise1,wavelet\_name,levels,thresh,'h',1);} \]

Here, \textit{noise1} is the image to be enhanced; \textit{wavelet\_name} is a character string containing the name of the wavelet; \textit{levels} is the depth of the decomposition pyramid; \textit{thresh} is the value of the threshold; and \textit{out} is the output image. Three other inputs stand for other filtering parameters which you need not worry about for this exercise.

Use wavelet thresholding to de-noise the noisy car image. Experiment with different wavelet families, filter orders, pyramid depths, and thresholds. To see the Matlab names for the different wavelets, try \texttt{help biorwavf} (for biorthogonal spline wavelets), \texttt{help dbwavf} (for Daubechies wavelets), \texttt{help symwavf} (for symmlets).

Turn in at least three different output images (for three different parameter settings), and comment on your results (what happens when you vary the filter order, the number of iterations, and the threshold?). For this example, reasonable values for the threshold are in the range 50-100. E.g., to use a Daubechies wavelet of order 2, have 7 filter iterations, and set the threshold at 50, use the following command:
out = wdencmp('gbl',noise1,'db2',7,50,'h',1);

(To find out more about wavelets, how to choose a wavelet, how to set the threshold, etc, take ECE 648!)