

Problem Set 1
CMSC 426
Due September 13, 2005

Written exercises (10 points each)

1. Find the maximum value, y , of the function $y = 3 + 2x - 4x^2$
2. Find the values of the following integrals:

a. $\int_0^{\pi} \sin(\theta) d\theta$

b. $\int_0^{\pi} \sin^2(\theta) d\theta$

c. $\int_0^{\pi} \sin(\theta) \cos(\theta) d\theta$

d. $\int_{\pi}^{2\pi} \sin(\theta) \cos(\theta) d\theta$

3. Inner Products

- a. Find the inner product between $u = (1, 2)$ and $v = (1, 3)$.
 - b. Use the inner product to find the angle between these two vectors.
 - c. Consider the vectors $u = (4, -1, 2)$ and $w = (2, y, 1)$. For what value of y are these vectors orthogonal?
 - d. Consider the vector $v = (1, -1, 3)$. Find two vectors, u and w , so that u , v , and w are all orthogonal to each other.
4. Suppose that u and v are two-dimensional vectors. Suppose further that $\langle u, v \rangle = 0$, $\|u\| = \|v\| = 1$. Let $w = (x, y)$. Show that:

$$x^2 + y^2 = (u \cdot w)^2 + (v \cdot w)^2$$

Programming Assignment (20 points each)

The goal of this assignment is to give you some practice with Matlab, and to better familiarize you with correlation by implementing it. In each of these problems, the number of loops that you are allowed to use (*for* or *while*) is strictly limited. This is to get you to start thinking about the Matlab style of implementation, in which we try to

replace loops by performing a single operation on a whole vector or array at once. This is much more efficient in Matlab than iteration, and often leads to more elegant programs. In each problem I will indicate the number of loops that I think are necessary. You will lose two points for every excess loop that you use. Hint: for replication, you may want to create small matrices that replicate values, and use these to pad your initial matrices. Check what happens when you perform matlab calls such as: `7*ones(4,4)` or `ones(3,1)*[5,2,4]`.

1. Implement a function to perform one-dimensional correlation. You should handle boundary pixels by replicating the pixels at the boundary (the second way of handling boundaries that is described in the class notes). Your function should therefore perform identically to the Matlab function *imfilter* when it is called with the 'replicate' option (see "help imfilter"). You are not allowed to use the function *imfilter*, or any related functions such as *convn* in your implementation. One loop is allowed.

Test your function by comparing the results of the following Matlab calls:

```
correlation_1d([1:4, 4:-1:1], [1/3 1/3 1/3])
imfilter([1:4, 4:-1:1], [1/3 1/3 1/3], 'replicate')
```

and

```
correlation_1d([1:4, 4:-1:1], .2*ones(1,5))
imfilter([1:4, 4:-1:1], .2*ones(1,5), 'replicate')
```

Turn in a file with your code, and a file showing the result of performing these operations.

2. In this problem, you will check the results of correlating an averaging filter with a sine wave.
 - a. Create a vector that is 100 elements long, so that the elements in it uniformly sample a sine wave, from 0 to 2π . That is, the vector should look like: $(\sin(0), \sin(2\pi/99), \dots, \sin(2\pi))$. No loops are allowed.

Use the functions *figure* and *plot* to display this vector. Use *print* to save the image in a JPEG file, and turn this in, along with the code used to generate it.

- b. Create an averaging filter that is 21 elements long, and correlate this with the sine wave you have generated. Using the functions above, and the additional command 'hold on', plot the result in the same figure as the original sine wave. Save this to a JPEG file, and turn this in along with code used to generate it. No loops are allowed.

3. Implement a two-dimensional correlation function. Again, boundary pixels should be handled with replication, and the results should be the same as those produced by *imfilter* with the '*replicate*' option.

Test your function by comparing the results of the following Matlab calls:

```
correlation_2d((1:5)'*(1:5), (1/9)*ones(3,3))  
imfilter((1:5)'*(1:5), (1/9)*ones(3,3), 'replicate')
```

and

```
correlation_2d((1:5)'*(1:5), (1/25)*ones(5,5))  
imfilter((1:5)'*(1:5), (1/25)*ones(5,5), 'replicate')
```

Turn in your code, and a file showing the results of these operations. Two loops are allowed.