# Problem Set 1
# CMSC 426
# Due Feb. 7
# 50 Points

This problem set will have a written part and a part that requires implementation in Matlab. Each written question is worth 5 points.

1. Suppose u and v are vectors, and ||u||=2, ||v||=3, and the angle between u and v is $\pi/4$. What is the inner product between u and v?

2. Suppose a 3D line passes through the origin and through the point (1,2,1). Where does this line intersect the plane: x+y-z=5?

3. Suppose $f(x,\theta) = \frac{\cos\theta}{x}$. What is $\partial f / \partial \Theta$?

4. If x is an eigenvector of both A and B then must it necessarily be an eigenvector of the matrix (AB)? Explain your reasoning

5. Suppose x is a random variable. P(x=1) = ½, P(x=0) = ¼, P(x=2) = ¼. What is the variance of x?

6. $\int\int(6xy^2 - 8xy + 3)dxdy = ??$

**Matlab Assignments**

You can use the Matlab function ps1, which we provide as a framework for writing and testing your code. Matlab has some special purpose functions that compute some of the operations below, but you should not use these. In particular, do not use Matlab functions `fspecial`, `imfilter`, `conv2`, `convn`, `filter2`. At the end of the assignment, we show the values produced by our code, which you may use to test your code. Include in your assignment a print-out of your code, and any figures generated by your code.

1. **15 points:** Write the function:

```
function If = mycorrelate(I, f)
```

This function takes two arguments, `I`, and `f`, which we can think of as an image and a filter. `I` will be a 2D matrix. `f` will be a 2D matrix also. (note that either one could also be 1D, in which case we can think of it as 2D, but with one dimension equal to 1). `If` will be the result of correlating `I` with `f`. Handle cases in which the

filter goes outside the boundary of the image by using 0 padding. That is, you can think of pixels outside the image as having the value of 0.

2. **10 points:** Write the function

```
function G = myGaussian(dims, sigma)
```

This function computes the value of a Gaussian distribution over a matrix of locations. These locations are based on the dimensions in `dims`. `G` contains the resulting values, so that `size(G)` will be equal to `dims`. For this problem, `dims(1)` will always equal 1, so you only need to compute the values of a 1D Gaussian distribution, in which the number of values you compute is equal to `dims(2)`. Assume the Gaussian has a mean of 0, and compute the function symmetrically about 0.

To show exactly what you should compute, let's take the example in which the function is called with: `myGaussian([1, 5], 1)`. In this case, `G(3)` will contain the value of a Gaussian with a standard deviation, `sigma`, of 1, at the location x = 0. `G(2)` will contain the value of the Gaussian at x = -1, while `G(4)` contains the value at x = 1. `G(1)` contains the value at x = -2, and `G(5)` contains the value at x = 2.

3. **5 points:** Notice in the previous problem that the contents of `G` do not necessarily sum to 1, even though the integral of the Gaussian is equal to 1. Can you explain why this is? Write a short explanation and hand it in. Now write a new function,

```
function Gn = myGaussian_normalized(dims, sigma)
```

This should produce the same values as `myGaussian`, but multiplied by a scale factor so that the contents of `Gn` sum to 1.

4. **Extra Credit (5 points):** Now extend your code so that it will generate the values of a 2D Gaussian function. In our version we use the same input arguments as in the 1D version. Specifically, there is only a single `sigma`. Assume that the Gaussian is circularly symmetric, so that a single `sigma` can describe its variance (technically, we say the Gaussian has a diagonal covariance matrix in which both diagonal elements equal sigma).

Here is the result of running `ps1` using our code:

```
>> ps1(1)

ans =

    1.7778    3.0000    3.6667    4.3333    3.1111
    4.3333    7.0000    8.0000    9.0000    6.3333
    7.6667   12.0000   13.0000   14.0000    9.6667
    6.2222    9.6667   10.3333   11.0000    7.5556

>> ps1(2)
```

```
ans =

    0.0044    0.0540    0.2420    0.3989    0.2420    0.0540    0.0044


ans =

    0.9997


ans =

    0.0000    0.0000    0.0000    3.9894    0.0000    0.0000    0.0000


ans =

    3.9894

>> ps1(3)

ans =

    0.0000    0.0000    0.0000    1.0000    0.0000    0.0000    0.0000


ans =

     1

>> ps1(4)

ans =

    0.0000    0.0000    0.0002    0.0000    0.0000
    0.0000    0.0113    0.0837    0.0113    0.0000
    0.0002    0.0837    0.6187    0.0837    0.0002
    0.0000    0.0113    0.0837    0.0113    0.0000
    0.0000    0.0000    0.0002    0.0000    0.0000
```