Some Matlab points as a result of recent questions

Creating column vectors

Consider the following statements.

```
a=[1 2 3 4 5 6];
B=[0  3 6;
   -1 0 5];
C=B';

c_a=a(:)
c_B=B(:)
c_C=C(:)
```

c_a, c_B and c_C are all 6×1 column vectors.

If x exists then

```
x=x(:)
```

gives a column vector.

Inner products and outer products

In MA2715 I mention the inner product of two vectors to create a scalar and the outer product which creates a matrix. This is what is done in Matlab with the following statements.

```
x=[1 3 5 7];
y=[2 4 6 8];

% inner product follows
v=x(:)'*y(:)

% outer product follows
A=x(:)*( y(:)' )
```

Entrywise operations and matrix operation

Consider the following statements.

```
A=[2 1;
   0 3]

for k=2:4
  E=A.^k
end

for k=2:4
  P=A^k
end
```

In the loop with `E=` the operation `.^` is the entrywise power.

In the loop with `P=` the operation `^` involves matrix multiplication to compute the power. We can only use `^` with matrices when we have square matrices.

Statements to complete functions

Suppose that a function file called `nearest_to_mean_pos.m` starts with the following statements. Give additional statements so that the output argument `k` is set to the position of the nearest entry of `x()` to the mean `mu`.

```
function [k, mu]=nearest_to_mean_pos(x)
n=length(x(:));
mu=mean(x(:));
k=0;
```

Suppose in a separate file you have the following.

```
a=[1 2 3 3.9 5 6];
[k, mu]=nearest_to_mean_pos(a)
```

If the function works correctly then you should get the following.

```
k =  4
mu =  3.4833
```

Statements to complete functions – another question

Suppose that a function file called `above_and_below.m` starts with the following statements. Give additional statements so that the output arguments `above`, `same` and `below` are set respectively to the number of entries which are greater than `mu`, are equal to `mu` and are less than `mu` in the vector `x()`.

```
function [above, same, below]=above_and_below(x)
n=length(x(:));
mu=mean(x(:));
above=0;
same=0;
below=0;
```

I suggest that you test your function.

"Advanced" answer

If you know about the function `min()` and how it can be used then the first task can be completed with one additional statement. With a slightly different name for the function you can have the following.

```
function [k, mu]=nearest_to_mean_pos2(x)
n=length(x(:));
mu=mean(x(:));
[~, k]=min(abs(x(:)-mu));
```

In a separate file you can check this with the following.

```
a=[1 2 3 3.9 5 6];
[k, mu]=nearest_to_mean_pos2(a)
```

Remarks about testing functions
Do not alter the input arguments

The assignment will not involve this but suppose that for 2 marks you had to create the following and test it.

```
function [k, mu]=nearest_to_mean_pos2(x)
n=length(x(:));
mu=mean(x(:));
[~, k]=min(abs(x(:)-mu));
```

You would get 0 marks if you submitted the following.

```
function [k, mu]=nearest_to_mean_pos2(x)
x=[1 2 3 3.9 5 6];
n=length(x(:));
mu=mean(x(:));
[~, k]=min(abs(x(:)-mu));
```

It is worthless as a function as it does not work for anything other than the vector `x` in the second line.

Remarks about testing functions
To minimally check for any syntax errors

As already indicated the following works.

```
function [k, mu]=nearest_to_mean_pos2(x)
n=length(x(:));
mu=mean(x(:));
[~, k]=min(abs(x(:)-mu));
```

Suppose that you remove a bracket in the last line to give the following.

```
function [k, mu]=nearest_to_mean_pos2(x)
n=length(x(:));
mu=mean(x(:));
[~, k]=min(abs(x(:)-mu);
```

If the last version is submitted then it would get 0 marks.

If you never use a function that you have just created then there is a reasonable chance that it may contain a syntax error.