ME2-HCPT Final Test Monday

Name:	CID number:

This is an open book exercise. You can **reuse/adapt/amend** any of your previous scripts, as long as you submit them.

Duration: 1h 20min

In your H drive create a folder *H:\ME2CPT\Final*.

Comment, sensibly, all your scripts **[5]**

Write at least a main script for each task, and name the files taskA.m, taskB.m, taskC.m. If you need or wish, you can then subdivide the tasks and write any associated functions, at your convenience.

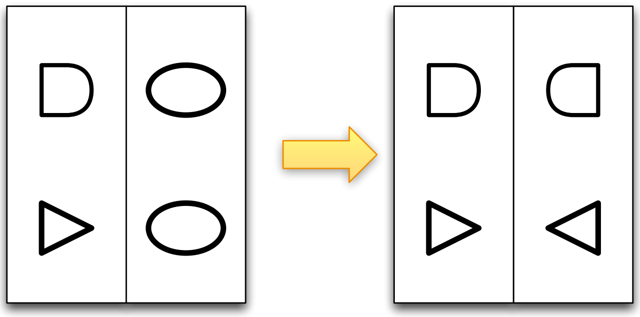
Task A [15]

The file *Obama.jpg* contains an image.

1. Read in the file and plot the image.

```
% read in the image file and store its content into matrix P
P = imread('Obama.jpg');
% plot the image
image(P)
```

2. Mirror the left half part of the image on to the right half part.



We need to know where the middle point is:

```
% enquire anout the size of the pictures, i.e. the number of pixels
N = size(P);
% get the central vertical axis
mid = ceil(N(2)/2);  N(2) is the number of column of array P. Reflect why
we need ceil and not floor: think of an odd number and where the middle
lays
```

We scroll from first column to middle column.
The first half remains unchanged. The second half is mirrored, starting the replacement from the end toward the middle, as we progress counting from the beginning to the middle.
Only the second index of P is involved.

```
% mirror the first half into the second half
for i = 1 : mid
    % scroll up to the middle vertical line and put every vertical line of
    % pixels into the second half, starting from the end.
    P(:,N(2)-i+1,:) =  P(:,i,:);
end
```

3. Render the left half part all blue and the right half part all red.

Get down the R and G for first half

```
% set the left half to blue
P(:,1:mid,1) = 0;
P(:,1:mid,2) = 0;
```

Get down the G and B for first half

```
% set the right half to red
P(:,mid+1:end,2) = 0;
P(:,mid+1:end,3) = 0;
```

4. Save the final image in the file *YesYouCan.jpg.*

```
% write the final image into a file
imwrite(P,'YesYouCan.jpg')
```

Task B [25]

The file *coefficients.txt* contains three pairs of data: $(x_a, a), (x_b, b), (x_c, c)$.
The values of $a(x)$, $b(x)$, and $c(x)$, serve as coefficients for the ordinary differential equation:

$$a(x)\frac{d^2y}{dx^2} + b(x)y + c(x) = 0$$

1. Fit the three sets of points individually with a fourth order polynomial.

Firstly, we read in the coefficients.

The text file contains a heading of characters, these have to be dismissed. Hence, we load the file from second line, first column, i.e. 1,0

```
% read the coefficients
A = dlmread('coefficients.txt',',',1,0);
```

We then extract the three pairs of data individually:

```
xa = A(:,1); ya = A(:,2);
xb = A(:,3); yb = A(:,4);
xc = A(:,5); yc = A(:,6);
```

We can now fit each of these with a poly of order 4

```
% fit the three sets of points with polynomials of order 4
c4a = polyfit(xa,ya,4);
c4b = polyfit(xb,yb,4);
c4c = polyfit(xc,yc,4);
```

2. Use the fitting polynomials to solve the differential equation in the range [0 : 0.01 : π], with boundary conditions $\left.\frac{dy}{dx}\right|_{x=0} = 3$ and $\left.\frac{dy}{dx}\right|_{x=\pi} = y$.

We need to set the domain of the ODE and evaluate the poly at the points of the domain. We do not need to evaluate the poly at their original x_a, x_b, x_c, as this is of little use for solving the ODE.

```
% define the interval for the ODE
h = 0.01;
x = [0:h:pi];
% evaluate the polynomials at the given interval, for the ODE
pa = polyval(c4a,x);
pb = polyval(c4b,x);
pc = polyval(c4c,x);
```

If we put the ODE in canonical form, as in tutorial 7, we can then reuse the same code. The format given was:

$$\frac{d^2y}{dx^2} + f(x)\frac{dy}{dx} + g(x)y = p(x)$$

```
% put the ODE in canonical form as defined in tutorial 7, so I can reuse
% the same discretised equations
% f = 0;
g = pb ./ pa;
p = -pc ./ pa;
```

Now we can use same code as in myodebc2

```
% discretise the ODE
% build a set of algebraic equation A y = b
N = length(x);
A = zeros(N,N);
```

Set the boundary conditions

Left

$\left.\frac{dy}{dx}\right|_{x=0} = 3$

This would be the equivalent of setting c1 = 1 and c2 = 0

```
% set the boundary conditions
% endpoint a: we need the forward scheme
```

```
A(1,1) = -1/h;
A(1,2) = 1/h;
b(1) = 3;
```

Right

This would be the equivalent of setting c3 = 1 and c4 = -1

$\left.\frac{dy}{dx}\right|_{x=\pi} = y$ aka $\left.\frac{dy}{dx}\right|_{x=\pi} - y = 0$

```
% endpoint b: we need the backward scheme
A(N,N-1) = -1/h;
A(N,N) = 1/h - 1;
b(N) = 0;
```

We can now set for the interior points.

Note that f in our case is = 0, hence it is omitted.

```
% discretise the interior points
for i = 2 : N-1
   % evaluate the function f, g and p at this x
   A(i,i-1) = 1/h^2; % - f(i) / (2*h);
   A(i,i) = g(i) - 2 / h^2;
   A(i,i+1) = 1/h^2; % + f(i) / (2*h);
   b(i) = p(i);
end
```

Finally solve the matrix:

```
% solve the equation
y = inv(A) * b';
```

3. Save the values of *x* and *y* in the file *myoutput.txt*, in two columns format.

Note that y is already in column format, but x is in a row format and needs to be transposed.

```
% save the results in a file
Sol = [x' y];
dlmwrite('myoutput.txt',Sol);
```

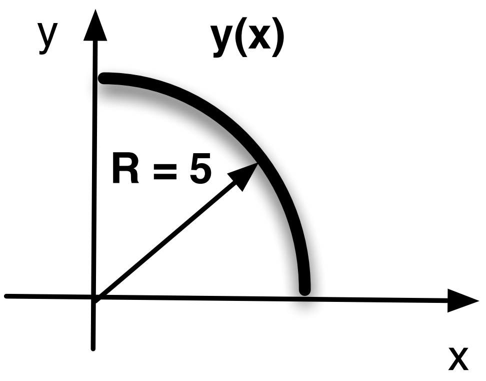
Task C [15]

1. Solve numerically, by using the trapezoidal scheme (with $dx = 0.01$), the integral:

$$I(x) = \int_0^x y(s)ds$$

in the range x = [0 : 0.01 : R].

The function y(x) is given in the figure:



2. Plot the numerical result $I(x)$ vs x in the given range.

Set the x and the y ranges:

```
% evaluate the function y(x) in the given range
x = [0:0.01:R];
y = sqrt(R^2 - x.^2);
```

For every value in the range of x, evaluate the integral along y

```
Nx = length(x);
% for every x, evaluate the integral
for j = 1 : Nx
    % compute the integral from 0 up to the current x
    I(j) = mytrapz(x(1:j),y(1:j));
end

plot(x,I,'o')
```

Upload ALL your scripts and results on Blackboard.

Useful Matlab functions for this test:

imread	- reads in data from image files
imwrite	- writes image data to a file
	- RGB: red, green, blue
dlmread	- reads multiple lines of numbers from a file
dlmwrite	- writes numerical data to a file
polyfit	- finds coefficients of polynomial to required degree
polyval	- evaluates a polynomial at specified points