The University of Western Ontario

Computer Science 2035b

Final Examination - Thursday, April 14th, 2016

Professor: John Barron

Last Name	
Given Names	
Student Number	

This exam consists of 10 questions (23 pages including this page) worth a total of 215 marks (which will be scaled to 100%). It is an open book exam, course notes and any MatLab book(s) are allowed. No calculators, laptops or cell phones are allowed. All answers are to be written in this booklet. Scrap work may be done on the back of each page; this will not be marked. The exam is 180 minutes long (3 hours) and comprises 35% of your final mark. Should your final exam grade be higher than your midterm exam grade (worth 20% of your final grade), your final exam grade in this course will count for the full 55% of your exam grade.

Please print you full name and student number, **as they appear on your student card**, in the space provided below before you start this exam.

(1)	20 marks		(6)	20 marks	
(2)	40 marks		(7)	15 marks	
(3)	15 marks		(8)	20 marks	
(4)	20 marks		(9)	30 marks	
(5)	20 marks		(10)	15 marks	

Total out of 215	

(1) (20 marks) Choose **one** answer (true or false) for each question.

(1) MatLab stands for **Mat**hmetics **Lab**atory. true [✓ false]

(2) The transpose of Matrices `(A*B)'` is `B'*A'`, where `'` is the MatLab transpose operator. [✓ true] false

(3) In the object-orient paradigm, **encryption** refers to storing the data and the methods (functions) that operate on them together. true [✓ false]

(4) For the 3 × 3 matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$, `mean(A)` yields $[\ 2 \quad 5 \quad 8\ ]$. [✓ true] false

(5) For $A = \begin{bmatrix} 5 & 2 \\ 2 & 5 \end{bmatrix}$, $A^{-1} = \frac{\begin{bmatrix} 5 & -2 \\ -2 & 5 \end{bmatrix}}{21}$ true [✓ false]

(6) A GUI built using GUIDE in MatLab has a fig file and an m file. [✓ true] false

(7) In object oriented MatLab, if class A ISA class B (A extends B) then class A can see all class B's private variables. true [✓ false]

(8) GPU solutions always run faster than single core solutions for calculations than are massively iterative, if overhead time is not considered. [✓ true] false

(9) Handle graphics always allows us to change the appearance of graphical entities that were plotted earlier, even if the handles for those plots are unknown. true [✓ false]

(10) Symbolic integration means MatLab can compute the definite integral of a function numerically. [✓ true] false

(11) GPU solutions always run faster than vectorized solutions on a single core if overhead time is not considered. true [✓ false]

(12) Numerical integration means MatLab can compute the definite integral of any function symbolically. true [✓ false]

(13) `randn(100,1)` yields a 100 component column vector of normal random numbers having a standard deviation of approximately $\sigma = 1.0$ and a mean of approximately $\mu = 1.0$. true [✓ false]

(14) A pixel is an edge intensity in an edge map. true [✓ false]

(15) A matrix with a condition number of 43.67 is almost singular. true [✓ false]

(16) Pixels with second order gradient values approximately equal to zero are probably edgels or near edgels. [✓ true] false

(17) MatLab 2015a can run the edge detector `'prewitt'` on the GPU. [✓ true] false

(18) Histogram equalization brightens dark parts of an image. [✓ true] false

(19) Histogram equalization darkens bright parts of an image. [✓ true] false

(20) Histogram equalization of a colour image requires that the colour and intensity information of the image be separated and that only the intensity information be histogram equalized. [✓ true] false

(2) (40 marks) Consider the polynomial $p(x) = x^5 * y^4 + x^2 * y$, where x and y can be set using the MatLab code:

```
lower_index=1;
upper_index=2;
num_indices=10;
x=linspace(lower_index,upper_index,num_indices);
y=linspace(lower_index,upper_index,num_indices);
```

(2a) (10 marks) Give the MatLab code for an efficient straightforward serial calculation of this polynomial. Print out the sum of the `p` values. Try to minimize the number of multiplications required.

Answer:

```
% serial solution
for i=1:num_indices
  x2=x(i)^2;
  x5=x2*x2*x(i);
  y2=y(i)*y(i);
  y4=y2*y2;
  p(i)=x5*y4+x2*y(i);
  end
fprintf('Serial sum: %f\n',sum(p(:)));
```

(2b) (10 marks) Give the MatLab code for an efficient straightforward vector calculation of this polynomial. Print out the sum of the `p` values. Try to minimize the number of multiplications required.

Answer:

```
% vector solution
  x2=x.^2;
  x5=x2.*x2.*x;
  y2=y.*y;
  y4=y2.*y2;
  p=x5.*y4+x2.*y;
fprintf('Vector sum: %f\n',sum(p(:)));
```

(2c) (10 marks) Give the MatLab code for an efficient straightforward multi-core calculation of this polynomial using the serial solution. Use 2 cores. Print out the sum of the `p` values. Try to minimize the number of multiplications required.

Answer:

```
% multicore solution
parpool('local',2);
parfor i=1:num_indices
  x2=x(i)^2;
  x5=x2*x2*x(i);
  y2=y(i)*y(i);
  y4=y2*y2;
  p(i)=x5*y4+x2*y(i);
  end
delete(gcp('nocreate'));
fprintf('parfor sum: %f\n',sum(p(:)));
```

(2d) (10 marks) Give the MatLab code for an efficient straightforward GPU calculation of this polynomial using the vectorized solution. Print out the sum of the `p` values. Try to minimize the number of multiplications required.

Answer:

```
% gpu solution
x=gpuArray(x);
y=gpuArray(y);
x2=x.^2;
x5=x2.*x2.*x;
y2=y.*y;
y4=y2.*y2;
p=x5.*y4+x2.*y
p=gather(p);
fprintf('gpu sum: %f\n',sum(p(:)));
```

(3) (15 marks) Consider the following 2D matrix:

```
A =  3     4   Inf
     5   NaN     6
   Inf     7   NaN
```

(3a) (8 marks) Give a MatLab code segment to compute the average and standard deviation, 5 ± 1.5811, of this matrix.

Answer:

```
A=A(~isnan(A(:)))
A=A(~isinf(A(:)))
% note that A is made a column vector from isnan
mean(A)
std(A)
```

(3b) (4 marks) Consider the following 2D matrix:

```
B =  3     4     5
     4     5     6
     5     6     7
```

What does the `mean(B)` print?

Answer:

```
mean(B)= 4     5     6
Note: these are the means of the 3 columns. The means of the 3 rows are
identical in this case.
```

(3c) (3 marks) What does the `mean(B(:))` print?

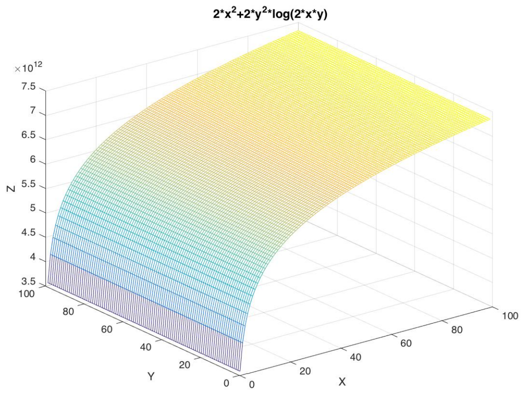
Answer:

```
mean(B(:))= 5
```

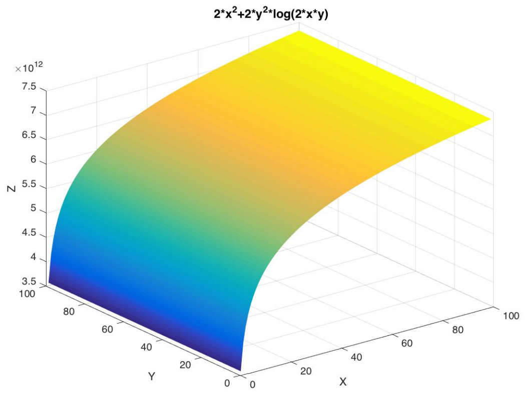
(4) (20 marks) Consider an initial 3D plots of the function

$$f(x, y) = 2 * x^2 + 2 * y^2 * log(2 * x * y)$$

for vectors `x=1:100` and `y=1:100`. Consider two 3D plots of this equation, shown in Figures 1a and 1b below.



(a)



(b)

Figure 1: (a) and (b) are the original 3D plots. These image are plotted independently of each other in latex and are not the result of a subplot command.

(4a) (10 marks) Give the matlab code to plot these 2 figures. Here the default MatLab colors, linewidths, axis tics, etc were used. The x, y and z labels have to be done (use fontsize 12). Be sure to set handles for the 2 figures for use in question (4b). Note the error in the figures' titles versus the actual function plotted. Figure 1a is plotted with `mesh` while Figure 1b is plotted with `surf` (with interpolated shading). To facilitate plotting these 2 functions you have been provided with the following MatLab code:

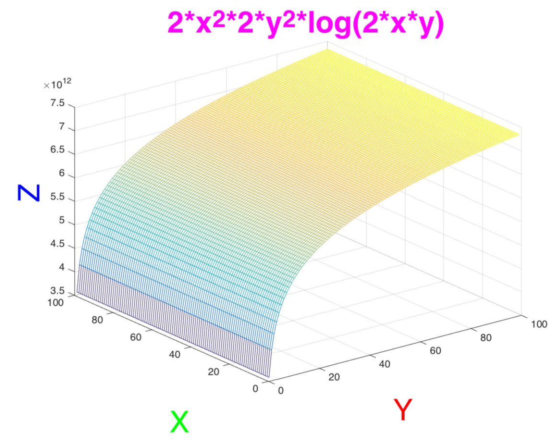
```
[X,Y]=meshgrid(x,y);
Z=2*X.^2.*2*Y.^2*log(2*X.*Y);
```

where `Z` is $f(x, y)$. `X`, `Y` and `Z` can be used the questions below. Please put your answer for this question on the next page.

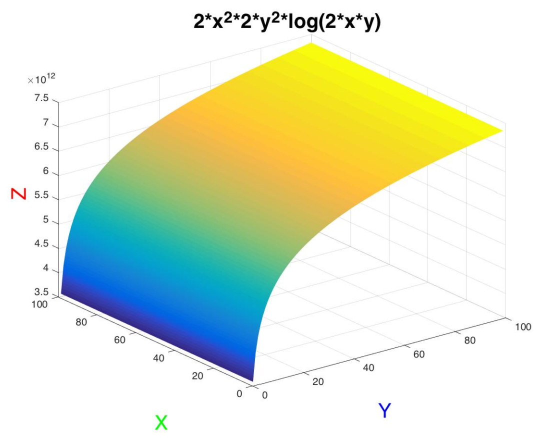
Answer:

```
h1=figure;

mesh(X,Y,Z);

xlabel('X');

ylabeil('Y');

zlabel('Z');

title('2*x^2+2*y^2*log(2*x*y)');


h2=figure;

surf(X,Y,Z);

shading interp

xlabel('X');

ylabel('Y');

zlabel('Z');

title('2*x^2+2*y^2*log(2*x*y)');


On 2016 exam I got the code which appears to work:

h1=figure;

m=mesh(X,Y,Z);

x1=xlabel('X');

y1=ylabel('Y');

z1=zlabel('Z');

t1=title('\fontsize{12} 2*x^2+2*y^2*log(2*x*y)');


h2=figure;

s=surf(X,Y,Z);

shading interp

x2=xlabel('X');

y2=ylabel('Y');

z2=zlabel('Z');

t2=title('\fontsize{12} 2*x^2+2*y^2*log(2*x*y)');
```

(4b) (10 marks) We want to change the appearance of the 3D plots using handle graphics. Figures 2a and 2b show the modified plots.



(a)



(b)

Figure 2: (a) and (b) are the modified plots.

We want to use the colours red, green and blue for the x, y and z labels respectively in Figure 1a and the colours blue, green and red for the x, y and z labels respectively in Figure 2b. The titles are now corrected and plotted in magenta and black. The labels and title in Figure 2a are to be printed with fontsize 30 while the labels and title in Figure 2b are to be printed with fontsize 20. Note that labels `X` and `Y` have been reversed. Your answers below should use handle graphics to do this and not re-make the plots from scratch. Do **not** use **gca** in your answer. Please put your answer for this question on the next page.

Answer:

```
figure(h1)
xlabel('\fontsize{30} \color{red} Y');
ylabel('\fontsize{30} \color{green} X');
zlabel('\fontsize{30} \color{blue} Z');
title('\fontsize{30} \color{magenta} 2*x^2*2*y^2*log(2*x*y)');

figure(h2)
xlabel('\fontsize{20} \color{blue} Y');
ylabel('\fontsize{20} \color{green} X');
zlabel('\fontsize{20} \color{red} Z');
title('\fontsize{20} \color{black} 2*x^2*2*y^2*log(2*x*y)');

% On the 2016 exam I got the code (which only works partially)
% Worth 7/10
figure(h1)
set(x1,'fontsize',30,'color','red')
set(y1,'fontsize',30,'color','green')
set(z1,'fontsize',30,'color','blue')
set(t1,'fontsize',30,'color','magenta');
% How to change the text? No title property for text
% The text does changes from 2*x^2+2*y^2*log(2*x*y)
% to 2*x^2*2*y^2*log(2*x*y) - the + changes to a *
% x and y labels get changed to Y and X - how to do now?

figure(h2)
set(x2,'fontsize',20,'color','blue')
set(y2,'fontsize',20,'color','green')
set(z2,'fontsize',20,'color','red')
set(t2,'fontsize',20,'color','black')
% How to change the text?
% Another way on the exam that also works for changing label
% and title colors and fonts but not actual text
% Worth 7/10
figure(h1)
set(gcf,'xlabel','Y','ylabel','X','AxesXColor','red',...
       'AxesYColor','green','AxesZColor','blue','TitleColor','magenta',...
       'TitleFontSizse',30);
figure(h2)
set(gcf,'xlabel','Y','ylabel','X','AxesXColor','blue',...
       'AxesYColor','green','AxesZColor','red','TitleColor','black',...
       'TitleFontSizse',20);
```

(5) (20 marks) Consider Figure 3 generated by the MatLab code below. Note that the *log* of negative numbers are not undefined but are complex numbers. The *log* of 0 is ∞. *abs*, of course, is the magnitude of both real and complex numbers.

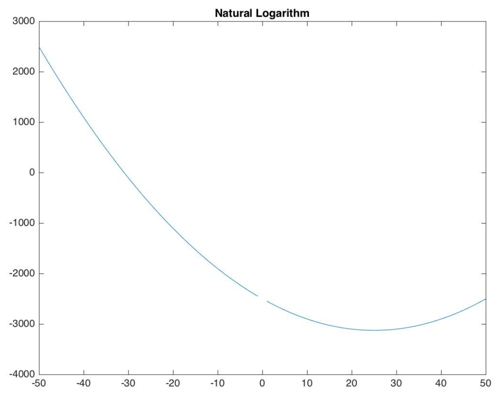


Figure 3: Simple plot of $f(x) = x^2 - 50 * x - abs(log(x)) - 2500$.

We can generate Figure 3 using the following MatLab code:

```
% natural logarithm ln(x) (base e)
x=-50:1:50;
y=x.^2-50*x-abs(log(x))-2500;
hf=figure
ha=plot(x,y)
title('Natural Logarithm');
```

(5a) (5 marks) Why is there a break in the plot at the $x = 0$ value?

Answer:

```
log(0) is inf (it is undefined) so a polyline segment can't be drawn using it.
Note that log(10) is 2.3026 while log(-10) is 2.3026+3.1416i
exp(log(10)) is 10 and exp(log(-10)) is -10, i.e.
exp(2.3026+3.1416i) is 10.
```

(5b) (5 marks) What would happen if `log` were replaced with `log10` or `log2` (base 10 and base 2 logarithms)?

Answer:

```
The infinity value at 0 still occurs, although the non-zero
values are still defined (and are a bit different than for log)
(numbers for log10 and log2 are real for x>0 and
complex for x<0, just as for log ==> so you get similar curves)
FYI:
log10(10) is  1.0000 while log10(-10) is 1.0000+1.3644i
log2(10)  is  3.3219 while log2(-10)  is  3.3219+4.5324i
```

(5c) (5 marks) Handles `hf` and `ha` have been set in the MatLab code provided above. Give the MatLab code using these handles to set the line color to red and the linewidth to 3 for the plot and set the background color to grey for the figure. Note that the grey figure background displays on the screen when this program is run but is not saved when using the `print` command. Hence, you don't see it in Figure 4 below.

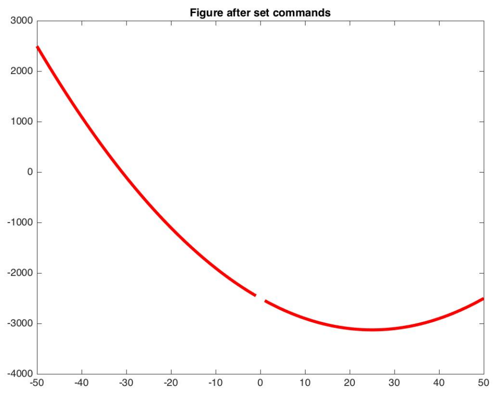


Figure 4: The function plot after the properties are changes via the figure and plot handle.

Answer:

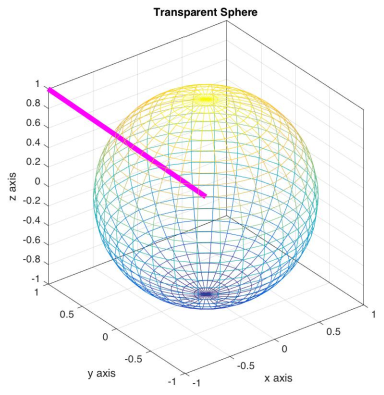
```
set(hf,'color',[0.5 0.5 0.5]);
set(ha,'color','red','linewidth',3.0);
```

(5d) (5 marks) Suppose `inspect(hf)` or `inspect(ha)` are been used. What would these commands do?

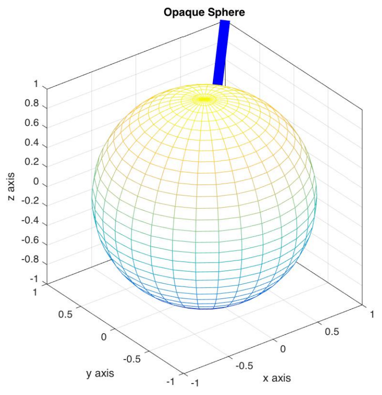
Answer:

```
Windows listing the properties and their current values for the
figure and axis (plot) objects will pop up. The user can
interactively change these properties. This is an alternative
to using handle graphics as above.
```

(6) (20 marks) Consider the two 3D plots shown in Figure 5.



(a)



(b)

Figure 5: Two 3D plots of a (a) transparent and a (b) opaque sphere. These image are plotted independently of each other in latex and are not the result of a subplot command.

(6a) (5 marks) Consider Figure 5a. This is a transparent sphere drawn using `sphere` with 30 latitude and longitude lines. It is plotted with a square axis. Turn the 3D box drawing on with 3D grid lines. Give the MatLab code to plot this transparent figure using all MatLab property defaults. Consider drawing polylines using `plot3` but other ways of doing this are also acceptable.

Answer:

```
figure
[X,Y,Z]=sphere(30); % or sphere(30,30)

% keeps the axis square and output the title and axis labels
% turn on the grid and 3D box
plot3(X,Y,Z)
axis('square')
title('Transparent Sphere')
xlabel('x axis')
ylabel('y axis')
zlabel('z axis')
grid on
```

```
box on

% some students suggested
m=mesh(x,y,z)
set(m,'facecolor','none')

mesh(x,y,z)
hidden off

Both work, but instead of lines you get transparent meshes.
```

(6b) (5 marks) Draw a line, with color magenta and linewidth 5, from the sphere center at (0,0,0) to the upper left hand corner (-1,1,1). Draw the line as a polyline (a collection of connected 3D points). Figure 5a shows this line with the transparent sphere. Note that the required code here is independent of the code in (6a).

Answer:

```
% the line goes from (-1, 1, 1) to (0, 0, 0)
hold on
a=0:-0.1:-1
b=0:0.1:1
c=0:0.1:1
% draw line
plot3(a,b,c,'color','magenta','linewidth',5);
```

(6c) (5 marks) Consider Figure 5b. This is an opaque sphere drawn using `sphere` with 30 latitude and longitude lines as in question (6a). It is also plotted with a square axis. Give the MatLab code to plot this opaque figure using all MatLab property defaults. This time, use `mesh` instead of a polyline to draw the surface and handle the opaqueness. Turn the 3D box drawing on with 3D grid lines.

Answer:

```
figure
[X,Y,Z]=sphere(30); % or sphere(30,30)

% keeps the axis square and output the title and axis labels
% turn on the grid and 3D box
mesh(X,Y,Z);
axis('square')
title('Opaque Sphere')
xlabel('x axis')
ylabel('y axis')
zlabel('z axis')
grid on
box on
```

(6d) (5 marks) Draw a line, with color blue and linewidth 10, from the sphere center at (0,0,0) to the upper right hand corner (1,1,1). Draw the line as a polyline (a collection of connected 3D points). Figure 5b shows this line with the opaque sphere. Note that the required code here is independent of the code in (6c).

Answer:

```
% the line goes from (0,0,0) to (1,1,1)
hold on
a=0:0.1:1;
b=0:0.1:1;
c=0:0.1:1;
% draw line
plot3(a,b,c,'color','blue','linewidth',10);
```

(7) (15 marks) Consider a simple edge detection scheme for colour images. This algorithm does **NOT** produce good results. Figure 6a shows the original lena.jpg image while Figure 6b shows its colour edge image.

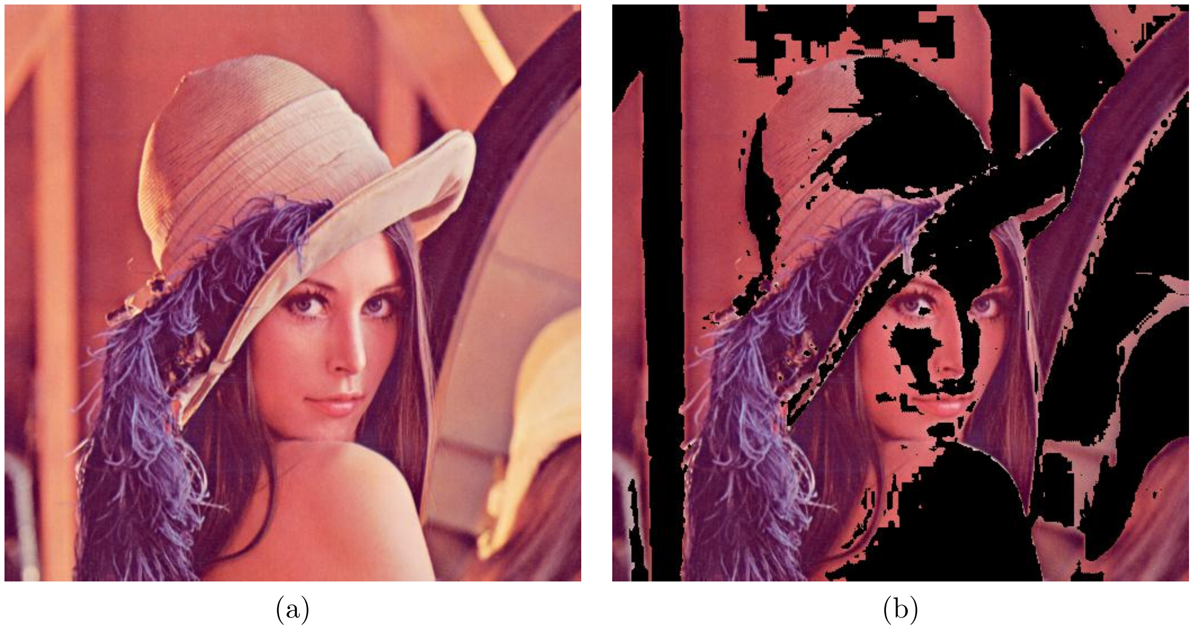


Figure 6: (a): original lena.jpg image and (b) its colour edge image

(7a) (5 marks) Show the MatLab code required to read lena.jpg, separate out the 3 colour planes for red, green and blue and compute the absolute value of the difference of the green and blue images as `diff`.

Answer:

```
RGB=imread('lena.jpg');
R=RGB(:,:,1);
G=RGB(:,:,2);
B=RGB(:,:,3);

figure, imshow(R,[]), title('Red')
figure, imshow(G,[]), title('Green')
figure, imshow(B,[]), title('Blue')

diff=abs(G-B);
```

(7b) (10 marks) Given `diff`, compute the edge map as all those locations where the difference is not 0. For these locations, set pixel values to be 0 in a colour image `edges`. For all other locations set `edges` to the corresponding colour values in lena.jpg.

Answer:

```
% set locations with non-zero differences to white
% Using a binary_edges image is not required for this question
binary_edges=zeros(size(RGB,1),size(RGB,2));
binary_edges((diff~=0))=0;   % diff is non-zero
binary_edges((diff==0))=255; % diff is zero

figure
imshow(binary_edges,[]);
title('Binary edges for green and blue plane differences')

colour_edges=RGB;
for i=1:512
for j=1:512
 % locations where G and B are different are made black
 if(binary_edges(i,j)==0) % or if(diff(i,j)~=0)
    colour_edges(i,j,1)=0;
    colour_edges(i,j,2)=0;
    colour_edges(i,j,3)=0;
    end % if
end % j
end % i
imwrite(colour_edges,'colour_edges_2016.jpg');
figure
imshow(colour_edges,[]);
title('Colour edges');
```

(8) (30 marks) The question concerns the Matlab symbolic arithmetic toolbox. Assume

$$f(x, y) = x^2 * y + y^2 * x$$

is both integrable and differentiable.

(8a) (5 marks) Write MatLab code to numerically evaluate this integral:

$$\int_{-\pi}^{\pi} \int_{-\frac{pi}{8}}^{-\frac{pi}{4}} f(x, y) dx dy.$$

Answer:

```
syms f x y
f=x^2*y+y^2+x
int(int(f,x,-pi,pi),y,-pi/8,-pi/4)
==>(x^2*y^2*(x + y))/6
eval(int(int(f,x,-pi,pi),y,-pi/8,-pi/4)) or
eval(int(int(f,y,-pi/8,-pi/4),x,-pi,pi))
In either case the answer by computer is 3.8937

Some students interpreted this question as numerical integration:
% need a meshgrid to compute X and Y, how about:
a=-pi/8:delta_a:-pi/4
b=-pi:delta_b:pi
% where delta_a is -pi/100 and delta_b is pi/100 say?
[X,Y]=meshgrid(a,b);
fun=@(X,Y)(X.^2.*Y+Y.^2.*X);
balue=integral2(fun,-pi/8,-pi/4,-pi,pi
```

(8b) (5 marks) Write MatLab code to numerically evaluate this derivative

$$\frac{\partial^2}{\partial x \partial y} f(x, y)$$

at $x = -\frac{\pi}{4}$ and $y = -\frac{\pi}{8}$.

Answer:

```
syms f x y
```

```
f=x^2*y+y^2*x

der=diff(diff(f,y),x)

==> der=2*x+2*y

x=pi, y=pi/8

eval(der)

The answer by computer is 7.0686
```

(8c) (10 marks) Consider the following MatLab code:

```
syms A AI a b c d e f g h i
A = [a b c ;
     d e f ;
     g h i];
AI=inv(A);
fprintf('The original A matrix:\n');
A
fprintf('A*AI');
simplify(A*AI)

fprintf('diag(A)\n');
diag(A)
fprintf('diag(A.*A)\n');
diag(A.*A)
```

What is printed?

Answer:

```
A = [ a, b, c]
    [ d, e, f]
    [ g, h, i]
A*AI= [ 1, 0, 0]
      [ 0, 1, 0]
      [ 0, 0, 1]
diag(A)
 a
 e
 i
diag(A.*A)
 a^2
 e^2
 i^2
```

(9) (30 marks) Consider the following MatLab code:

```
n = 1000;
A = zeros(n,n,'double');
A(400,404) = 1.0;
A(678,2) = 8.0;
A(678,999) = 4.0;
A(705,678) = 2.0;
A(705,999) = 10.0;
w = ones(n,1);
v = A*w;
```

Note that `A` only has 5 non-zero elements. Answer the following questions.
(9a) (5 marks) What is the value of `v(678)`? Hint: to answer this question consider the dot product of `w` with the appropriate row of A.
Answer:

```
The dot product of row 678 with w yields 8*1+4*1=12
i.e. A(678,2) is 8 and A(678,999) is 4.
```

(9b) (5 marks) What is the value of `v(999)`?
Answer:

```
v(999)=0
i.e. note that row 999 is all 0's (column 999 has two non-zero
values at A(678,999) and A(705,999 but these have not effect
on the requited dot product calculation)
```

(9c) (5 marks) How many bytes does `A` use?
Answer:

```
1000*1000*8=8000000 bytes or 8MB (as a double requires 8 bytes)
```

(9d) (5 marks) If `A=sparse(A)` is computed then typing `A` in the command window yields what?
Answer:

```
A=
(678,2)         8
(400,404)       1
(705,678)       2
(678,999)       4
(705,999)      10
```

(9e) (5 marks) A sparse array stores one index (represented as a double) for each column in the array, one row index and the value for each non-zero element in the array and one additional double number (unknown) that specifies a larger maximum value of non-zero values (for example `A` has only 5 non-zero values but the maximum number of additional non-zero values can be considerably larger than this). This additional number allows you to add more non-zero elements to `A` without having to reallocate the sparse matrix.
How many bytes does sparse `A` require?. This is the value returned by `whos A`.
Answer:

```
There are 1000 column indices ==> 8000 bytes as indices are double.
There are 5 non-zero values so we need 5 row indices and 5 values
for a total of 80 bytes ==> 8080 bytes.
The addition maximum number of non-zero values requires 8 bytes
==> 8088 bytes in total. This is the value returned by "whos A".
```

(9f) (5 marks) If the size of `v` computed as `v=A*w` using non-sparse `A` is 8000 bytes (1000 elements with each element being 8 bytes) what is the size of `v` computed as `v=A*w` for sparse `A`?
Answer:

```
v is 1000*8 = 8000 bytes still (even though A is sparse, it has
this sparseness has no effect on the size of v.)
```

(10) (15 marks) This question concerns animation as done on assignment 3 and lab 6 of this year. Consider the function:

$$f(x,y) = x^2 * (1 + cos(\theta)) + y^2 * (1 + cos(\theta)).$$

Figure 7 shows the images at $\theta = 0°$ (the same as $\theta = 360°$) and at $\theta = 180°$.

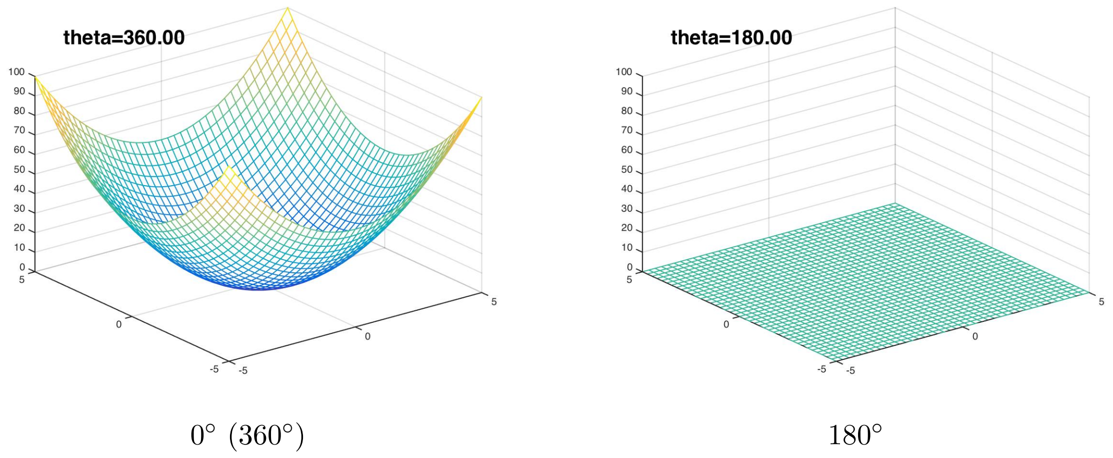


Figure 7: Animation images for 0° (same as 360°) and 180°.

(10a) (5 marks) What is the vectorized MatLab statement to compute $f(x,y)$ in MatLab variable `Z`? Assume that θ is in degrees and note that `cos` requires arguments in radians.
Answer:

```
Z=X.^2.*(1+cos(theta*2*pi/360))+Y.^2.*(1+cos(theta*2*pi/360));
or
Z=(X.^2+Y.^2).*(1+cos(theta*2*pi/360))
```

(10b) (5 marks) The animation is to be generated by varying θ from 0 to 360° by some $\delta\theta$ step. What is the MatLab `for` statement that would allow you to compute $f(x,y)$ for 101 θ values? [Hint: note that the 1st and 101th values are the same and compute $\delta\theta$.]
Answer:

```
for theta=0:360/100:360 % 101 thata values
or % another way
for theta=linspace(0,360,101)
```

(10c) (5 marks) Assume that `x` and `y` vary from -5 to 5. How would you compute the arguments for an `axis` command so that each of the 100 images displayed use the same 3D axis? Obviously `xmin` and `ymin` are -5 and `xmax` and `ymax` are 5/ Obviously `zmin` is 0. What should `zmax` be so that the entire surface is displayed for all valid `x` and `y` values? Explain your reasoning: an answer without an explanation is worth nothing.

Answer:

```
The maximum x^2 value is 25
The maximum y^2 value is 25
The maximum cos(theta) value is 1 (theta=0)
Thus, using the function, we have
zmax=25*(1+1)+25*(1+1)=100;
```