Appendix

1 Question 4

The plots are attached below:

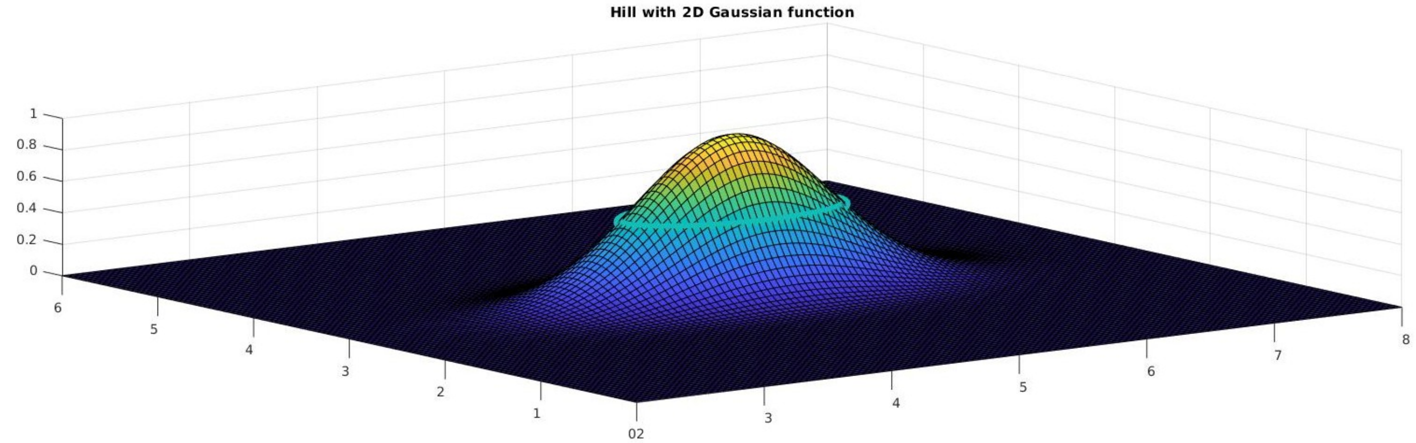

Figure 1: Hill constructed using 2D gaussian function

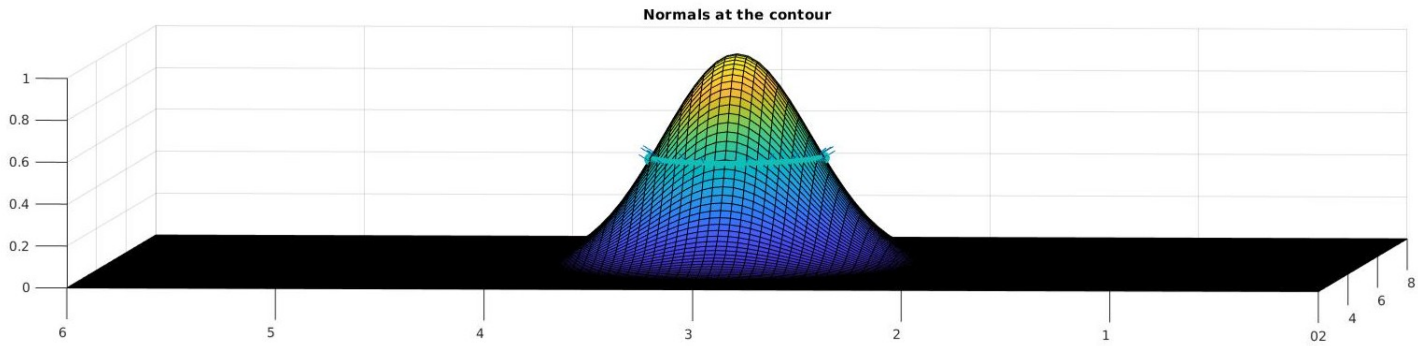

Figure 2: Normals plotted at the contour

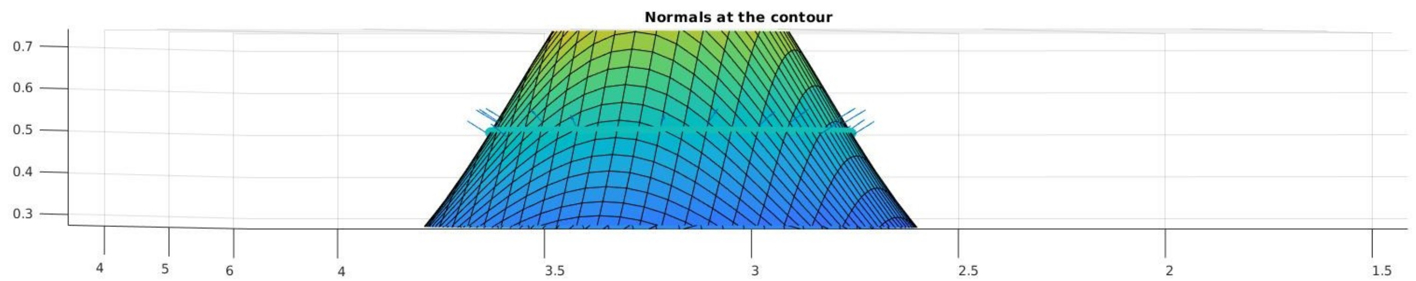

Figure 3: Zoomed in view to see the normals

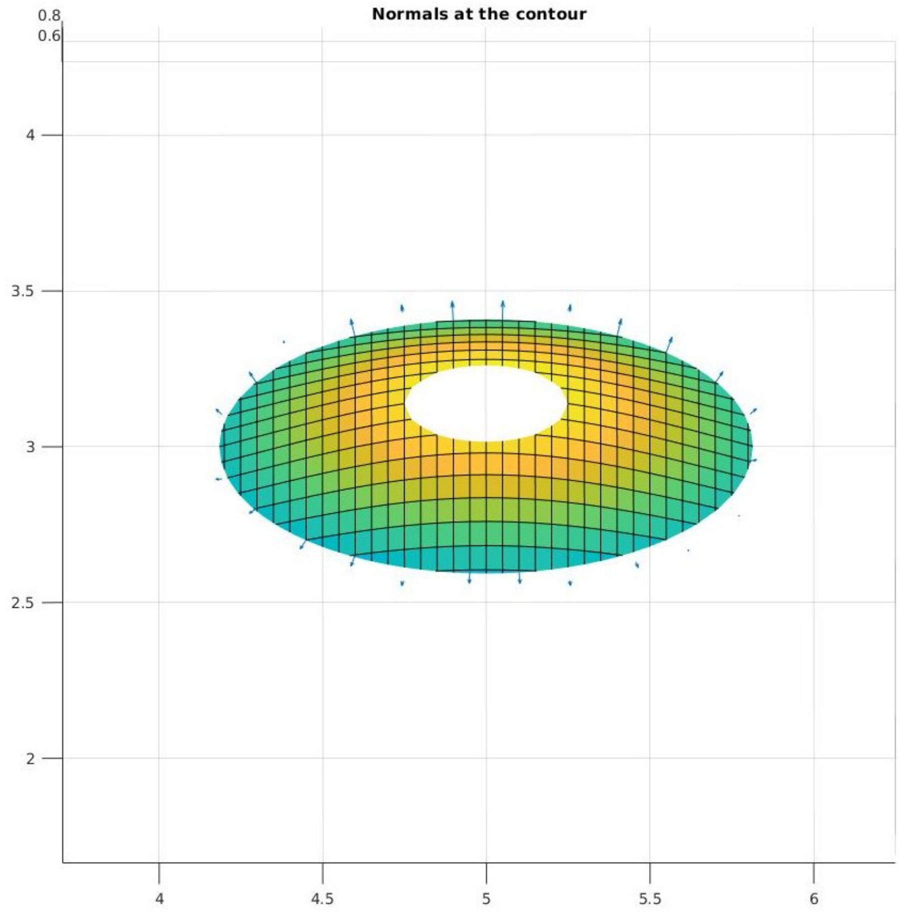

Figure 4: Top View to see the equispaced contours

Note: I have used the external function, **curvspace** in MATLAB to analytically compute equispaced points on the chosen contour and then plot normals at those points. ref: Curvspace in MATLAB file exchange

1.1 Code used

```
w = 0.05;   %discretisation parameter
x = 2:w:8;  %range for x and y
y = 0:w:6;

[X,Y] = meshgrid(x,y);

Z = exp( -(X-5).^2 - 4.*(Y-3).^2 );   %construct 2d gaussian hill
z_max = max(max(Z));
z_contour = z_max/2;    %find contour at half the height

figure(1);
surf(X,Y,Z);    %plotting the 2d gaussian hill
title("Hill with 2D Gaussian function");
hold on;
[~,H] = contour3(X,Y,Z,[z_contour,z_contour]);    %marking contour at half  the height
H.LineWidth = 8;
axis equal;
hold off;

figure(2);
surf(X,Y,Z);    %plotting the 2d gaussian hill
title("Normals at the contour");
hold on;
```

```
[C,H] = contour3(X,Y,Z,[z_contour,z_contour]);    %plot contour
H.LineWidth = 8;

P = C(:,2:101)';
Eq_P = curvspace(P,25)';    %find equispaced points on the contour

[nx,ny,nz] = find_normal(Z,w);    %defined function to calculate normal
filter = filterxy(X,Y,Eq_P);
quiver3(X, Y, Z, nx .* filter, ny .* filter, nz .* filter);    % plotting normals at equispaced points
axis equal;
hold off;

%{
% Validation for finding the normal done using the inbuilt surfnorm function
figure(3);
surf(X,Y,Z);
title("Validation: Normals found from surfnom");
hold on;
[~,C] = contour3(X,Y,Z,[z_contour,z_contour]);
C.LineWidth = 8;
[U,V,W] = surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W);
axis equal;
hold off;
%}

function filter = filterxy(X,Y,q)    %to overlay equispaced points on the predefined mesh
    tol = 0.026;
    filter = zeros(size(X));
    %calculate nearest points on the mesh for each one of the equispaced
    %points. then do elementwise OR for individual filters to get net filter
    for i=1:size(q')
        temp_filter = X < (q(1,i) + tol) & X > (q(1,i) - tol) & Y < (q(2,i) + tol) & Y > (q(2,i) - tol);
        filter = filter | temp_filter;
    end
end

function [nx,ny,nz] =  find_normal(g,w)    %to find normal for a surface g
    %g = z - f(x,y),  normal(g) = 1 - grad(f)
    [nx,ny] =  gradient(g .* (-1/w));
    nz = ones([1 + (6/w), 1 + (6/w)]);
end
```

.

2 Question 5

Explanation mentioned in handwritten solution. Obtained expression for H is

$$\vec{H} = \frac{j}{\omega\mu}(x(cos(z) - z^3)\hat{y} \tag{1}$$

2.1 Code used

```
syms x y z w mu;     %define symbols for x,y,z, permiability

E = [x * sin(z), y^2, z^3 * x];     %give input electric field expression here
%mu = 4*pi*(10^(-7));     %give permeability value here

H = symbolic_H_finder(E, mu)

function H = symbolic_H_finder(E, mu)
    syms x y z w;
    Ex = E(1); Ey = E(2); Ez = E(3);     %define x,y,z components of E

    %curl(E) = -jwu*H,     H = (j/wu) * curl(E)
    Hx = ( diff(Ez,y) - diff(Ey,z) ) * 1i / (w * mu) ;     %assign corresponding curl elements to components of H
    Hy = ( diff(Ex,z) - diff(Ez,x) ) * 1i / (w * mu) ;
    Hz = ( diff(Ey,x) - diff(Ex,y) ) * 1i / (w * mu) ;

    H = [Hx, Hy, Hz];
end
```