ANEXO A LA PRACTICA Nº 1 DE CAMPOS

1. **Fundamentos teóricos sobre funciones de Matlab.**

1.1 *Función "contour":* Dibuja el contorno de las curvas de nivel de un vector "V" según los valores que irán tomando la pareja de coordenadas cartesianas escogidas (pueden formarse todas las combinaciones: X-Y, X-Z ó Y-Z).

Así por ejemplo, contour (x,y,V) representaría el contorno del vector V (quien tendrá dependencia respecto a todas o sólo algunas de las coordenadas cartesianas), tomando el plano X-Y como referencia para representar automáticamente varias curvas. Si se quisiera controlar la cantidad de líneas de contorno a representar podría añadirse como parámetro añadido a esta función la cantidad de curvas a representar, quedando de esta forma la sintaxis de la función: contour(x,y,V,3), siendo 3 el número de curvas a ser representadas.

1.2 *Función "meshgrid":*

Esta función es necesaria para complementar a la función descrita anteriormente porque es necesario especificar el rango de valores a ser usado por las variables cartesianas sobre cuyos ejes representaremos el contorno de un determinado vector.

Por ejemplo en el caso de representar el vector V sobre el plano X-Y usando la función "contour", deberíamos crearnos previamente una malla de valores x-y mediante un barrido de ambas variables a lo largo de un rango, así podríamos generarnos un rango de valores para x-y:

```
[x,y] = meshgrid (-2:.2:2, -2:.2:2);
```

Sin embargo de esta forma estaríamos limitando el uso de la función "contour" al plano X-Y, cuando podríamos tener la necesidad de hacer uso de la representación del vector V sobre otro plano distinto. Para que esta situación no se produzca se establecerá un barrido por las tres variables cartesianas: x,y,z de la siguiente forma:

```
[x,y,z] = meshgrid (-2:.2:2, -2:.2:2, -2:.2:2);
```

donde estamos tomando valores de las tres variables en pasos de 0.2 en 0.2 desde el punto (-2,-2,-2) hasta el punto (2,2,2).

Así de esta forma podríamos escoger con la función "contour" sobre qué plano representar las curvas de nivel: X-Y, Y-Z ó X-Z.

1.3 *Función "gradient"*:

[px,py,pz] = gradient (V) devuelve el gradiente númerico de V. Siendo px la derivada parcial de V respecto a x, py respecto a y, pz respecto a z. Si no se añade ningún otro parámetro más a la función "gradient" se asumirá que el espacio de separación entre puntos en cada una de las tres direcciones analizadas será uno. Si se quisiera cambiar el espaciamiento entre puntos podríamos darle ese espaciamiento como parámetro de la función para cada una de las tres variables:

```
[px,py,pz] =gradient (V,.2,.2,.2);
```

1.4 *Función "quiver"*:

quiver(x,y,px,py) con esta función podemos dibujar pequeñas flechas dirigidas según los valores del vector p = (px,py), el cual podríamos por ejemplo obtener usando la función gradiente explicada en el punto anterior, en los puntos x,y, los cuales son los parámetros iniciales de esta función "quiver". Esta función automáticamente escala las flechas en función de la cuadrícula que se genera para visualizar una figura.

Esta función podría servir para representar el campo eléctrico generado a partir de un potencial dado, ya que tomando la expresión del potencial en función de las variables x,y podríamos calcular su gradiente con la función "gradient", y con el vector "p" generado representar sobre el mismo dibujo al mismo tiempo las curvas equipotenciales y el campo eléctrico.

1.5 *Función "spline"*:

Esta función realiza la interpolación a partir de unos pares de puntos (x,y), de forma que crea a partir de ellos unos nuevos pares (x1,y1) los cuales serán los puntos a usar en la interpolación. Es necesario tener creado el rango de valores para x1 antes de hacer uso de la función "spline". La sintaxis de esta función queda como sigue:

```
y1 = spline (x,y,x1);
```

1.6 *Función "Plot"*:

Con esta función de Matlab podremos visualizar expresiones del estilo y = f(x), donde "x" e "y" serán normalmente vectores, con un rango de valores especificado según el ejercicio. Esta

función tiene una gran versatilidad en cuanto al uso de los parámetros porque directamente podrá admitir sólo dos parámetros (x,y), o bien podremos especificar que tipo de representación tendrán dichos puntos dentro de una gama de posibilidades que podemos encontrar en la ayuda de esta función si tecleamos: "help plot". Con lo cual podríamos emplear hasta tres parámetros.
Pero también existe la posibilidad de representar al mismo tiempo varios grupos de estos parámetros definidos hasta ahora, tal y como se refleja en la sintaxis:

```
plot(x1,y1,s1,x2,y2,s2,x3,y3,s3,...)
```

donde s1,s2,s3,..., etc., son los "string" que definirán con que tipo de punto estaremos representando a la expresión y=f(x).
Un ejemplo del uso de la función "plot" con dos grupos de parámetros lo tenemos más adelante en la representación de la onda senoidal.

1.7 ***Función "int"***:

Con esta función se podrá realizar una integral definida entre dos límites, dado que así la necesitaremos en el cálculo de la energía almacenada en una determinada distribución de corriente o carga. Usa tres parámetros: "S" será la expresión que pretendemos integrar, "a" y "b" son los límites de la integral definida, la sintaxis de esta función quedará:

Resultado=int(S,a,b); donde Resultado contendrá el valor calculado por "int".
Resultado=int(S,v,a,b); en este caso calculamos la integral de "S" respecto de la variable "v" entre los límites "a" y "b". Si la función "S" tiene más variables que "v", deben ser definidas como simbólicas mediante en comando syms:

```
syms v x a b;
S=cos(v)+sin(x);
Resultado=int(S,v,a,b)
```

1.8 ***Nota***: en cualquier caso, existen herramientas de consulta en esta versión de Matlab que permitirán completar las funciones anteriormente citadas o cualquier otra que se utilice en los ejemplos que a continuación aparecen. Para poder hacer dichas consultas bastará con escribir en la "Command Window" la

palabra "help" con lo cual aparecerán todos los grupos de funciones que contiene esta versión de Matlab, o bien podemos entrar en un entorno de ayuda bajo windows tecleando "helpwin". También existe la posibilidad de correr demostraciones ya creadas en forma de librerías si tecleamos "demo".

2. **Ejemplo para representar una función potencial y el campo eléctrico asociado.**

En primer lugar genero los valores a emplear en el mallado para las variables x e y, observando que se genera una cantidad distinta de valores para la x que para la y:

```
[x,y] = meshgrid(-2:.2:2,-1:.15:1);
```

A continuación planteo la ecuación correspondiente al potencial en función de las variables x e y:

```
V= x .* exp(-x.^2 - y.^2);
```

A partir del potencial V, genero un operador gradiente:

```
[px,py] = gradient(V,.2,.15);
```

Represento el potencial V y mantengo dicha figura en pantalla:

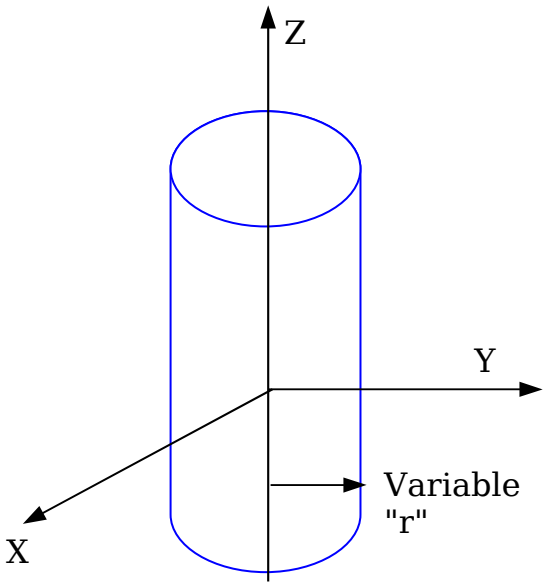
```
contour(x,y,V), hold on
```

Con la figura del potencial en pantalla represento el operador gradiente negado (puesto que el campo eléctrico se dirige hacia potenciales cada vez más pequeños, es decir, justo lo contrario que hace el gradiente), sobre un rango de valores para la "x" y la "y"

```
quiver(x,y,-px,-py), hold off, axis image
```

3. **Línea uniformemente cargada.**

La línea infinita se colocará sobre el eje Z, y las superficies equipotenciales se generarán con simetría cilíndrica en torno a ella



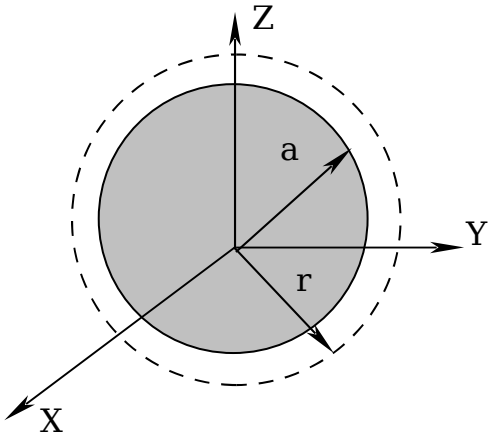
3.1Aplicación de la función "contour" sobre diferentes combinaciones de ejes cartesianos :

```
%Línea infinita distribuida sobre el eje z
%El valor introducido por teclado debe ser el nombre del fichero

%Valor de la densidad lineal de carga (C/m)


d=1e-9;

%defino la constante dieléctrica "e" en el vacío y le asigno su %valor

e=1/(4*pi*9*10^9);


%defino tres variables "x", "y", "z" para usarlas en la
%posterior visualización hecha con "contour"

%Corte con el plano (x,y)

[x,y] = meshgrid(-2:.2:2, -2:.2:2);
z=0;


%Según el estudio teórico el cálculo del potencial eléctrico "V"
% dependerá del logaritmo de la distancia a la línea, por tanto
% si considero que la línea está sobre el eje cartesiano Z, tendré
% que la distancia a dicha línea dependerá de la raíz cuadrada del
% cuadrado de los pares de puntos (x,y) que haya tomado
```

```
V = d*log(sqrt(x.^2+y.^2))/(2*pi*e);


%puede visualizarse las lineas equipotenciales usando el plano X-Y
%usando un mismo radio para todas las líneas equipotenciales visibles
%desde este plano. Vamos a representar 5 superficies
%equipotenciales (5 radios distintos) cortadas sobre el plano X-Y

contour(x,y,V,5);


%O también se puede visualizar las lineas equipotenciales usando el
%plano X-Z. También representamos 5 superficies equipotenciales.

[x,z] = meshgrid(-2:.2:2, -2:.2:2);
y=0;
V = d*log(sqrt(x.^2+y.^2))/(2*pi*e);
contour(x,z,V,5);


%También puede visualizarse las lineas equipotenciales usando el
%plano Y-Z. Escriba la parte de programa correspondiente en
%este caso.
```

Nota: Visualizar por separado las equipotenciales, ejecutando tres veces el programa, en cada ejecución visualizar un solo "contour", por ejemplo forzando comentarios sobre los dos contornos que no se quiera ver. Observar cómo se generan las líneas equipotenciales y razonar si existe algún tipo de simetría. Otra alternativa es usar la función "subplot" para almacenar varias gráficas en una misma representación. Para conocer el uso de la función "subplot" puede tomar información haciendo "help subplot" dentro de la ventana de comandos de Matlab.

4. **Esfera uniformemente cargada (potencial, campo eléctrico).**



4.1 *Representación del campo eléctrico en función del parámetro radial (r)* :

```
%El valor introducido por teclado debe ser el nombre del fichero

%Valor del radio de la esfera (m)

A=5e-3;

%Valor de la densidad de carga volumétrica (C/m^3)

p=1e-9;

%defino la constante dieléctrica "e" en el vacío y le asigno su %valor

e=1/(4*pi*9*10^9);

%se definen dos rangos de valores para representar los valores
%dentro y fuera de la esfera cargada. Con "r1" y "r2" se
%representa a la variable radial, medida desde el centro de la esfera
%utilizamos el comando linspace para hacer el barrido en la coordenada
%radial.

r1=linspace(0,A,20);
r2=linspace(A,5*A,20);

%"E1" se utiliza para representar los valores dentro, mientras
%"E2" para los valores fuera, del campo eléctrico.

E1=p*r1/(3*e);
E2=p*A^3./(3*e*r2.^2);

%Finalmente se representan simultáneamente los resultados obtenidos
%para el campo eléctrico tanto dentro como fuera de la esfera


 plot(r1,E1,'b',r2,E2,'b');
 grid;
```

4.2 *Representación de superficies equipotenciales y del campo eléctrico sobre éstas usando diferentes ejes cartesianos*:

```
%El valor introducido por teclado debe ser el nombre del fichero
```

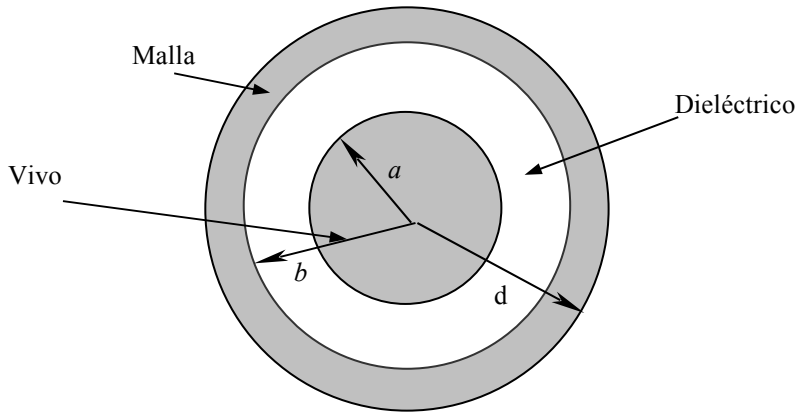
```
%Los parámetros y variables usados para esta función coinciden con
%los usados para la función esfera del apartado anterior.

A=5e-3;
p=1.0e-9;
e=1/(4*pi*9*10^9);


% Corte según el eje z=0; representación en el plano (x,y).

[x,y] = meshgrid(-A:A/20:A, -A:A/20:A);
z=0;

%Las curvas equipotenciales en el interior quedarían descritas por la
%siguiente expresión de "V1" (potencial eléctrico) respecto a un
%punto cualquiera (x,y,z), considerando que la esfera está centrada
%en el origen de coordenadas cartesianas.

 rr=sqrt(x.^2+y.^2+z^2);

 % Cogemos los indices para ver que posiciones están dentro de la
 % esfera y que posiciones están fuera.
 Ind_fuera=(rr>A);
 Ind_dentro=(rr<=A);

 % Calculamos el potencial dentro y fuera de la esfera.
 % Primero, definimos una matriz de ceros.
 V_dentro=zeros(size(rr));
 V_fuera=zeros(size(rr));
 % Después, calculamos los valores del potencial en sus posiciones.
 V_dentro(Ind_dentro)=(p/(2*e)).*(A^2-rr(Ind_dentro).^2/3);
 V_fuera(Ind_fuera)=p.*A^3./(3*e.*rr(Ind_fuera));
 % Finalmente, unimos ambas contribuciones (dentro y fuera).
 V1=V_dentro+V_fuera;

%creamos el gradiente de este potencial eléctrico en las tres
%direcciones según los ejes cartesianos.


 [px,py] =gradient(V1,A/20,A/20);

%puede visualizarse las líneas equipotenciales usando los ejes "x" e
%"y", representando las flechas la dirección del campo eléctrico, por
%esta razón se coloca el signo menos a los vectores resultantes del
%gradiente (puesto que el campo apunta hacia potenciales cada vez
%menores)
```

```
contour(x,y,V1,1),hold on, quiver(x,y,-px,-py), hold off

%puede visualizarse las lineas equipotenciales usando los ejes "z" e
%"y"

[y,z] = meshgrid(-A:A/20:A, -A:A/20:A);
x=0;

rr=sqrt(x^2+y.^2+z.^2);

V_dentro=zeros(size(rr));
V_fuera=zeros(size(rr));
V_dentro(Ind_dentro)=(p/(2*e)).*(A^2-rr(Ind_dentro).^2/3);
V_fuera(Ind_fuera)=p.*A^3./(3*e.*rr(Ind_fuera));
V1=V_dentro+V_fuera;

[py,pz] =gradient(V1,A/20,A/20);

contour(y,z,V1,1),hold on,

quiver(y,z,-py,-pz), hold off

grid;

%puede visualizarse las lineas equipotenciales usando los ejes "x" y
%"z". Escribir vosotros como sería el programa para este último caso.
```

Nota: Al igual que ocurría con el cable cargado de longitud infinita, se debe visualizar por separado las equipotenciales, ejecutando tres veces el programa, en cada ejecución visualizar un solo "contour", por ejemplo forzando comentarios sobre los dos contornos que no se quiera ver. Observar cómo se generan las potenciales y razonar si existe algún tipo de simetría. También puede utilizar el comando "subplot", o "figure" para almacenar las tres gráficas en una misma representación, o añadir nuevas ventanas cada una con su gráfica.

5. **Coaxial (intensidad de campo magnético, potencial, energía en un punto interior del vivo, inductancia).**

Corte en sección



5.1 *Representación del campo eléctrico en función de la distancia al centro del coaxial*:

Usar el siguiente programa grabado en un fichero (.m).

```
%El valor introducido por teclado debe ser el nombre del fichero

%Datos del problema

I=1e-3;
a=5e-3;
b=4*a;
d=6*a;

%u es la constante de permeabilidad magnética en el vacío

u=4*pi*10^(-7)

%densidad de corriente dentro del vivo

p=I/(pi*a^2);

%densidad de corriente en la malla

q=-I/(pi*(d^2-b^2));

%se definen dos rangos de valores para representar los valores del
%radio, variable que mide las distancias desde el centro del coaxial
%para las distintas regiones

r1=linspace(0,a,20);  %para el interior del vivo
r2=linspace(a,b,20);  %para la zona del dieléctrico
r3=linspace(b,d,20);  %para la zona de la malla
```

```
%H1 se utiliza para representar los valores dentro del vivo, H2 para
%el dieléctrico
%y H3 para los valores en la malla


H1=p*r1/2;
H2=I./(2*pi*r2);
H3=-q*(d^2-r3.^2)./(2*r3);


%Visualizamos todos los tramos

plot(r1,H1,'b',r2,H2,'b',r3,H3,'b');

grid;
```

5.2 *Cálculo de la Energía magnética y de la autoinducción por unidad de longitud*:

Usar el siguiente programa.

```
%Calculo la energía magnética de este sistema calculando la integral
%del campo magnético, dentro de un volumen cilíndrico que encierre un
%metro de longitud del cable coaxial

%En primer lugar hacemos el cálculo simbólico para tener
%la inductancia de forma analítica.
%Defino todas las variables como simbólicas porque así las
%necesita interpretar la función de Matlab "int"(integración)

syms pi a b d u I r1 r2 r3;

%Cálculo de la autoinductancia del cable interior

w1=2*pi*(u/2)*int(r1*((I*r1)/(2*pi*a^2))^2,r1,0,a);

L1=2*w1/(I^2)

%Cálculo de la inductancia mutua

w2=2*pi*(u/2)*int(r2*(I/(2*pi*r2))^2,r2,a,b);

L2=2*w2/(I^2)

%Cálculo de la autoinductancia del cable exterior
```

```
w3=2*pi*(u/2)*int(r3*(I*(d^2-r3^2)/(2*pi*r3*(d^2-b^2)))^2,r3,b,d);

L3=2*w3/(I^2)

%Repetir estos cálculos pero dando los valores del problema a todos
%los parámetros que aparecen, con el fin de tener el valor
%numérico de la inductancia
```