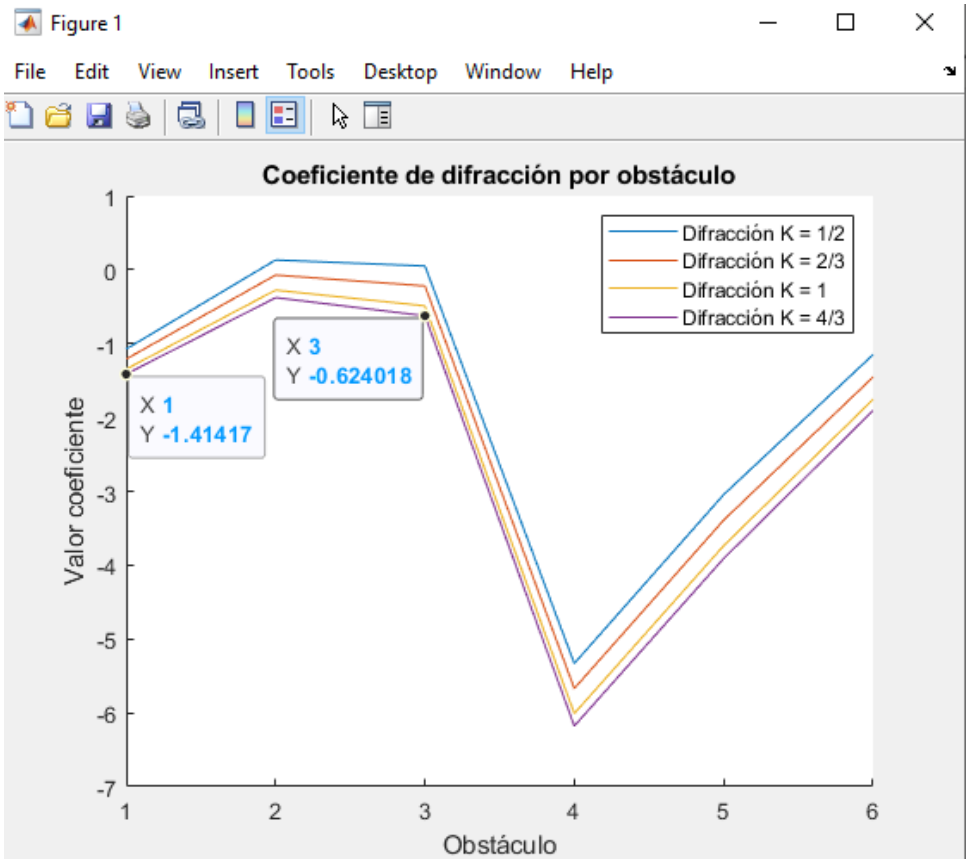
Tarjeta 2.1: **Calcular de forma iterativa el parámetro de difracción del enlace completo y de los subvanos derecho e izquierdo para los valores de k propuestos en el ejercicio 2.**

Para el cálculo de los parámetros de difracción, hemos realizado un bucle for que vaya calculando en función de los valores de "K" el parámetro de difracción para cada obstáculo. Código está ideado de tal forma que haya una iteración por cada valor de "K", aunque no está pensado para casos de vectores de "K" con más de una fila.

De esta forma, obtenemos la gráfica inferior, que representa en el eje X los obstáculos del primero al sexto, en el eje Y el propio valor del coeficiente de difracción, y 4 curvas pertenecientes a cada valor de "K".

De forma adicional, se ha marcado el coeficiente de difracción de los subvanos derecho e izquierdo, que se identifican por ser los mayores presentes de entre todos los posibles a la derecha e izquierda del obstáculo principal. Como se puede observar, siempre van a ser esos dos obstáculos los subvanos, pese a cambiar la "K".



Captura del código incompleto:

```
clear;close all;clc;

f      = 5800e6;
c      = 3e8;
lambda = c/f;
d      = 20.09e3; %en Km
R0     = 6370e3;

e  = [797 800 803 799 735 760 788 805];
a  = [10 0 0 0 0 0 0  8];
d1 = [0 0.806e3 1.910e3 3.721e3 7.831e3 10.955e3 14.965e3 d];
d2 = d - d1;

K           = [0.5, 2/3 ,1, 4/3];
Re          = R0*K;
altura_rayo = ((e(end)+a(end)-e(1)-a(1))/d)*d1 + e(1)+a(1);
R1          = sqrt(lambda*d1.*d2/d);
d2=d-d1;

obstaculo_mayor          = max(e);
posicion_obstaculo_mayor = find(e==obstaculo_mayor);
numero_iteraciones       = size(Re);
columnas                 = size(d1);

figure(1);title("Coeficiente de difracción por obstáculo");
for iteracion=1:numero_iteraciones(2)
    flecha_iterada(iteracion,:)        = (d1.*d2)/(2*Re(iteracion));
    despejamiento_iterado(iteracion,:) =  e + flecha_iterada(iteracion,:) - altura_rayo;
    uve_iterado(iteracion,:)           = sqrt(2)*despejamiento_iterado(iteracion,:)./R1;

    Ldif_iterado(iteracion,:)          =  6.9 + 20*log10(sqrt((uve_iterado(iteracion,:)-0.1).^2 +1) + uve_iterado(

    uve_obstaculo_principal_y_subvano(iteracion,:)  = [uve_iterado(iteracion,posicion_obstaculo_mayor-1),uve_itera
    posicion_uve_menos_negativo_inferior_subvano    = find( (uve_iterado(iteracion,:))<(min(uve_obstaculo_principa

    hold on
    plot(uve_iterado(iteracion,:))
    xticks(1:6);
    hold off

end
ylabel("Valor coeficiente");xlabel("Obstáculo");
```