Anexo I.

1. Código en Scilab de la función Toa (tiempos de llegada).

```
function x=Toa(t0,t,x,y)

//fusión de datos tiempo de llegada

//BSi=[xi,yi];

r=(t-t0)*3e8 ;

K=sqrt(x.^2+y.^2);
xa=x(2:length(x));
ya=y(2:length(y));
H=[xa' ya'];
r1=r(1);
Ka=K(2:length(K));
ra=r(2:length(r));
b=0.5*[(Ka.^2)'-(ra.^2)'+(r1^2)];
x=inv(H'*H)*(H')*b;

endfunction
```

2. Código en Scilab de la función Tdoa (diferencia de tiempos de llegada).

```
function z=Tdoa(t0,t,x,y)

//fusión de datos diferencia de tiempo de llegada

//BSi=[xi,yi];

r=(t-t0)*3e8-(t(1)-t0)*3e8;

K=sqrt(x.^2+y.^2);
xa=x(2:length(x));
ya=y(2:length(y));
H=[xa' ya'];
Ka=K(2:length(K));
ra=r(2:length(r));
c=-ra';
d=0.5*[(Ka.^2)'-(ra.^2)'];
z=inv(H'*H)*(H')*(t(1)*3e8*c+d);



endfunction
```

3. Código en Scilab de la función Aoa (ángulo de llegada).

```
function z=Aoa(t0,n,x,y,a)

//BSi=[xi,yi];

H=[-tan(a(1)) 1];
for i=2:n
  h=[-tan(a(i)) 1];
  H=[H;h];
end
B=[-tan(a(1))*x(1)+y(1)];
for j=2:n
  b=-tan(a(j))*x(j)+y(j);
B=[B;b];
end
z=inv(H'*H)*(H')*B;

endfunction
```

4. Código en Scilab de la función Hib (híbrido de Toa y Aoa).

```
function z=Hib(t0,n,t,x,y,a,pond)

//fusión de datos ángulo de llegada
//BSi=[xi,yi];

Haoa=[-tan(a(1)) 1];
for i=2:n
  haoa=[-tan(a(i)) 1];
  Haoa=[Haoa;haoa];
end
Baoa=[-tan(a(1))*x(1)+y(1)];
for j=2:n
  baoa=-tan(a(j))*x(j)+y(j);
Baoa=[Baoa;baoa];
end
zaoa=inv(Haoa'*Haoa)*(Haoa')*Baoa;

//fusión de datos tiempo de llegada

r=(t-t0)*3e8 ;
K=sqrt(x.^2+y.^2) ;
xa=x(2:length(x));
ya=y(2:length(y));
Htoa=[xa' ya'];
r1=r(1);
Ka=K(2:length(K));
ra=r(2:length(r));
btoa=0.5*[(Ka.^2)'-(ra.^2)'+(r1^2)];
ztoa=inv(Htoa'*Htoa)*(Htoa')*btoa;

// mezcla

z=pond*ztoa+(1-pond)*zaoa;

endfunction
```

5. Código en Scilab de la función Potencia (estima de la posición con almacenamiento previo del retardo).

```
function qest=potencia

//La función devuelve el punto de la fuente emisora

N=8; //número de antenas
T=24; //retrasos de 0 a 24 (24 posibles retrasos y localizaciones)

vector=zeros(N,T);

vector(1,1)=3000; vector(2,2)=3000; vector(3,3)=3000; vector(4,4)=3000;
vector(5,5)=3000; vector(6,6)=3000; vector(7,7)=3000; vector(8,8)=3000;
vector(1,2)=3162; vector(2,1)=3162; vector(2,3)=3162; vector(3,2)=3162;
vector(3,4)=3162; vector(4,3)=3162; vector(4,5)=3162; vector(5,4)=3162;
vector(5,6)=3162; vector(6,5)=3162; vector(6,7)=3162; vector(7,6)=3162;
vector(7,8)=3162; vector(8,7)=3162;
vector(1,3)=3605; vector(2,4)=3605; vector(3,5)=3605; vector(3,1)=3605;
vector(4,2)=3605; vector(4,6)=3605; vector(5,3)=3605; vector(5,7)=3605;
vector(6,4)=3605; vector(6,8)=3605; vector(7,5)=3605; vector(8,6)=3605;
vector(1,4)=4242; vector(2,5)=4242; vector(3,6)=4242; vector(4,1)=4242;
vector(4,7)=4242; vector(5,2)=4242; vector(5,8)=4242; vector(6,3)=4242;
vector(7,4)=4242; vector(8,5)=4242;
vector(1,5)=5000; vector(2,6)=5000; vector(3,7)=5000; vector(4,8)=5000;
vector(5,1)=5000; vector(6,2)=5000; vector(7,3)=5000; vector(8,4)=5000;
vector(1,6)=5831; vector(2,7)=5831; vector(3,8)=5831; vector(6,1)=5831;
vector(7,2)=5831; vector(8,3)=5831;
vector(1,7)=6708; vector(2,8)=6708; vector(7,1)=6708; vector(8,2)=6708;
vector(1,8)=7615; vector(8,1)=7615;
vector(1,9)=2000; vector(2,10)=2000; vector(3,11)=2000; vector(4,12)=2000;
vector(5,13)=2000; vector(6,14)=2000; vector(7,15)=2000; vector(8,16)=2000;
vector(1,10)=2236; vector(2,9)=2236; vector(2,11)=2236; vector(3,10)=2236;
vector(3,12)=2236; vector(4,11)=2236; vector(4,13)=2236; vector(5,12)=2236;
vector(5,14)=2236; vector(6,13)=2236; vector(6,15)=2236; vector(7,14)=2236;
vector(7,16)=2236; vector(8,15)=2236;
vector(1,11)=2828; vector(2,12)=2828; vector(3,9)=2828; vector(3,13)=2828;
vector(4,10)=2828; vector(4,14)=2828; vector(5,11)=2828; vector(5,15)=2828;
```

```
vector(6,12)=2828; vector(6,16)=2828; vector(7,13)=2828; vector(8,14)=2828;
vector(1,12)=3605; vector(2,13)=3605; vector(3,14)=3605; vector(4,9)=3605;
vector(4,15)=3605; vector(5,10)=3605; vector(5,16)=3605; vector(6,11)=3605;
vector(7,12)=3605; vector(8,13)=3605;
vector(1,13)=4472; vector(2,14)=4472; vector(3,15)=4472; vector(4,16)=4472;
vector(5,9)=4472; vector(6,10)=4472; vector(7,11)=4472; vector(8,12)=4472;
vector(1,14)=5385; vector(2,15)=5385; vector(3,16)=5385; vector(6,9)=5385;
vector(7,10)=5385; vector(8,11)=5385;
vector(1,15)=6324; vector(2,16)=6324; vector(7,9)=6324; vector(8,10)=6324;
vector(1,16)=7280; vector(8,9)=7280;
vector(1,17)=1000; vector(2,18)=1000; vector(3,19)=1000; vector(4,20)=1000;
vector(5,21)=1000; vector(6,22)=1000; vector(7,23)=1000; vector(8,24)=1000;
vector(1,18)=1414; vector(2,17)=1414; vector(2,19)=1414; vector(3,18)=1414;
vector(3,20)=1414; vector(4,19)=1414; vector(4,21)=1414; vector(5,20)=1414;
vector(5,22)=1414; vector(6,21)=1414; vector(6,23)=1414; vector(7,22)=1414;
vector(7,24)=1414; vector(7,24)=1414; vector(8,23)=1414;
vector(1,19)=2236; vector(2,20)=2236; vector(3,17)=2236; vector(3,21)=2236;
vector(4,18)=2236; vector(4,22)=2236; vector(5,19)=2236; vector(5,23)=2236;
vector(6,20)=2236; vector(6,24)=2236; vector(7,21)=2236; vector(8,22)=2236;
vector(1,20)=3162; vector(2,21)=3162; vector(3,22)=3162; vector(4,17)=3162;
vector(4,23)=3162; vector(5,18)=3162; vector(5,24)=3162; vector(6,19)=3162;
vector(7,20)=3162; vector(8,21)=3162;
vector(1,21)=4123; vector(2,22)=4123; vector(3,23)=4123; vector(4,24)=4123;
vector(5,17)=4123; vector(6,18)=4123; vector(7,19)=4123; vector(8,20)=4123;
vector(1,22)=5099; vector(2,23)=5099; vector(3,24)=5099; vector(6,17)=5099;
vector(7,18)=5099; vector(8,19)=5099;
vector(1,23)=6082; vector(2,24)=6082; vector(7,17)=6082; vector(8,18)=6082;
vector(1,24)=7071; vector(8,17)=7071;

z=rand(1,7616,"normal")+%i*rand(1,7616,"normal");
x=zeros(N,7616);

for n=1:N
   for m=1:vector(n,24)
      x(n,m)=0;
   end
   for m=vector(n,24)+1:7616
```

```
        x(n,m)=z(1,m-vector(n,24));
    end
end

//Suponemos diferentes localizaciones: vector será un tiempo
//diferente para cada localización dentro de cada antena

q=zeros(1,24);

for i=1:24
    q(i)=i;
end

y=0;
potant=0;

for m=1:T
    for n=1:N
        t1=vector(n,m);
        for t2=1:7616-t1
            xa(n,t2)=x(n,t2+t1);
        end
        for t3=7616-t1:7616
            xa(n,t3)=0;
        end
        y=y+xa(n,:);
    end

    pot=sum(abs(y.^2));

    if pot>potant
        potant=pot;
        qest=q(m);
    end
    y=0;
end
endfunction
```

6. Código en Scilab de la función prueba3 (fusión de datos en condiciones NLOS).

```
function[r1est,r2est,r3est,a1est,a2est,a3est]=prueba3(a1,a2,a3,r1,r2,r3,n)

//n es el número de rayos que llegan a la BS
//probar con: prueba3(37,18,16,510,632,728,40) el último puede ser 10,
//200...
//tomamos n rayos con errores en distancia y ángulo

r1barra=r1+(r1/10)*(rand(1,n));
r2barra=r2+(r2/10)*(rand(1,n));
r3barra=r3+(r3/10)*(rand(1,n));

a1barra=a1+(a1/10)*rand(1,n,["normal"]);
a2barra=a2+(a2/10)*rand(1,n,["normal"]);
a3barra=a3+(a3/10)*rand(1,n,["normal"]);



varr1=stdev(r1barra);
varr2=stdev(r2barra);
varr3=stdev(r3barra);
vara1=stdev(a1barra);
vara2=stdev(a2barra);
vara3=stdev(a3barra);

//tomamos el primer rayo de entre los que nos llegan y cogemos la distancia y el
ángulo

r11=r1barra(1);
r22=r2barra(1);
r33=r3barra(1);

a11=a1barra(1);
a22=a2barra(1);
a33=a3barra(1);
```

```
//he optado por coger la mínima entre las distancias porque se supone que las que llegan siempre van a ser mayores a la real, y un ángulo que sea media de los que llegan porque aquí no hay restricción (puede venir de cualquier parte, sobre todo si rebota)

mr1=min(r1barra);
mr2=min(r2barra);
mr3=min(r3barra);

ma1=mean(a1barra);
ma2=mean(a2barra);
ma3=mean(a3barra);


//esto son datos iniciales del problema a resolver

valor=0;
valant=100000000;
r12=860;
r23=1300;
r31=1063;
theta1=84.3;
theta2=54.5;
theta3=41.2;
alfa=0.15;

//las variables a obtener son j1, j2, j3, i1, i2, i3//
//j es la distancia e i es el ángulo//
//dentro del bucle if están las restricciones de distancia y de ángulo//

for j1=mr1-2*varr1:1:mr1
   for i1=ma1-vara1:0.5:ma1+vara1
      for j2=mr2-2*varr2:1:mr2
         for i2=ma2-vara2:0.5:ma2+vara2
            for j3=mr3-2*varr3:1:mr3
               for i3=ma3-vara3:0.5:ma3+vara3
                  if ((j1^2+j2^2-2*j1*j2*cos(%pi-(i1+i2)*%pi/180))<(1+alfa)*(r12^2)) & ((1-alfa)*(r12^2)<(j1^2+j2^2-2*j1*j2*cos(%pi-(i1+i2)*%pi/180))) & ((j2^2+j3^2-2*j2*j3*cos(%pi-(i3+(theta2-i2))*%pi/180))<(1+alfa)*(r23^2)) & ((1-
```

```
alfa)*(r23^2)<(j2^2+j3^2-2*j2*j3*cos(%pi-(i3+(theta2-i2))*%pi/180))) & ((j3^2+j1^2-
2*j3*j1*cos(%pi-(theta3-i3)*%pi/180-(theta1-i1)*%pi/180))<(1+alfa)*(r31^2)) & ((1-
alfa)*(r31^2)<(j3^2+j1^2-2*j3*j1*cos(%pi-(theta3-i3)*%pi/180-(theta1-i1)*%pi/180)))
                    valor=((r11-j1)^2)/(varr1^2)+((a11-i1)^2)/(vara1^2);
                    valor=valor+((r22-j2)^2)/(varr2^2)+((a22-i2)^2)/(vara2^2);
                    valor=valor+((r33-j3)^2)/(varr3^2)+((a33-i3)^2)/(vara3^2);
                    if valor<valant
                       valant=valor;
                       r1est=j1;
                       a1est=i1;
                       r2est=j2;
                       a2est=i2;
                       r3est=j3;
                       a3est=i3;
                    end
                 end
                 valor=0;
              end
           end
        end
     end
  end
end

endfunction
```