# Anexo II.

## 1. Código en Scilab de la función TOA con modificación en la red y en el terminal (estima del tiempo de llegada).

```
function t=TOA(retv,K)

//Método TOA

//fD=60*1e3
//To=1.12*1e-9
//El retraso podría estar dado en milésimas de segundo


fD=60*1e3;
To=1.12*1e-9;

//v es una señal gaussiana
//h varía sólo entre 0 y 1
//s(n) la hemos supuesto gaussiana, con diferentes valores de potencia

//s=5*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));
//s=10*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));
s=20*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));

//valor de A

A=0.5;


//cálculo de Nopt con el método 1

contador=besselj(0,2*%pi*To*fD);
i=2;

while abs(contador)<0.1 & i<K
```

```
    contador=contador+besselj(0,2*%pi*i*To*fD);
    i=i+1;
end

//Nopt=i;

//cálculo de Nopt con el método 2

//Nopt=5;
Nopt=50;


M=floor(K/Nopt);
Kopt=M*Nopt;
J=0;
Jant=0;

v=rand(1,Kopt,["normal"])+%i*rand(1,Kopt,["normal"]);
//v=zeros(1,K)+%i*zeros(1,K);
//h=ones(1,K);
//h=rand(1,K)+%i*rand(1,K);

sr=zeros(1,Kopt)+%i*zeros(1,Kopt);

//cálculo de s(n) retardada

for n=1:Kopt-retv
    sr(1,n)=s(1,n+retv);
end

for n=Kopt-retv+1:1:Kopt
    sr(1,n)=0;
end


//esto serviría para calcular una secuencia h que variara sólo en cada intervalo
```

```
h=zeros(1,Kopt)+%i*zeros(1,Kopt);

for k=0:M-1
    x=rand(1,["uniform"]);
    for ki=1:Nopt
        h(1,ki+k*Nopt)=x+%i*x;
    end
end

//cálculo de la secuencia recibida

r=A*(h.*sr)+v;

//cálculo del retraso

for ret=0:1:50 //retraso entre 0 y un número que creamos conveniente
    sret=zeros(1,K);
    for n=1:1:K-ret
        sret(1,n)=s(1,n+ret);
    end
    for n=K-ret+1:1:K
        sret(1,n)=0;
    end
    for m=1:1:M
        for n=(m-1)*Nopt+1:1:m*Nopt
            J=J+r(1,n)*conj(sret(1,n));
    end
    end
    //if J>Jant
    if (real(J)>real(Jant))&(imag(J)>imag(Jant))
        Jant=J;
        t=ret;
    end
    J=0;
end

endfunction
```

**2. Código en Scilab de la función TOAarray con modificación en la red y en el terminal (estima del tiempo de llegada con ayuda de antenas inteligentes: Método 1).**

```
function resultado=TOAarray

//datos iniciales del problema

Na=5;
K=50;
alpha=35;
d=0.5;
lambda=0.5;
retraso=49;
X=100;

JML=0;
JMLanterior=1e3;

//Cálculo de s,v,h y a

//s=5*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));
//s=10*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));
s=20*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));

v=rand(Na,K,["normal"])+%i*rand(Na,K,["normal"]);

a=1;
for na=1:1:Na-1
a=[a;cos(2*%pi*d/lambda*na*sin(alpha*%pi/180))+%i*sin(2*%pi*d/lambda*na*sin(alpha
*%pi/180))];
end


//h=0.9*(ones(1,K)+j*ones(1,K));
//h=0.5*(ones(1,K)+j*ones(1,K));
h=rand(1,K)+%i*rand(1,K);
```

```
//cálculo de s retrasada

sr=zeros(1,K);

for n=1:1:K-retraso
   sr(n)=s(n+retraso);
end

for n=K-retraso+1:1:K
   sr(n)=0;
end

//cálculo del vector recibido r

r=a*(sr.*h)+v;

//cálculo del retraso

for ret=0:X
   sret=zeros(1,K);
   for n=1:1:K-ret
      sret(n)=s(n+ret);
   end
   for n=K-ret+1:1:K
      sret(n)=0;
   end

   JML=norm(r-a*(sret.*h));

   if JML<JMLanterior
      JMLanterior=JML;
      resultado=ret;
   end
   JML=0;
end

endfunction
```

**3. Código en Scilab de la función TOAarray2 con modificación en la red y en el terminal (estima del tiempo de llegada con ayuda de antenas inteligentes: Método 2.).**

```
function t=TOAarray2

//datos iniciales del problema

fD=60*1e3;
To=1.12*1e-9;
Na=5;
K=50;
alpha=35;
d=0.5;
lambda=0.5;
retraso=40;
X=100;

//cálculo de s,v y a

//s = 5*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));
//s = 10*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));
s = 20*(rand(1,K,["normal"])+%i*rand(1,K,["normal"]));

v = rand(Na,K,["normal"])+%i*rand(Na,K,["normal"]);

a=1;

for na=1:1:Na-1

a=[a;cos(2*%pi*d/lambda*na*sin(alpha*%pi/180))+%i*sin(2*%pi*d/lambda*na*sin(alpha
*%pi/180))];
end


//cálculo de Nopt con el método 1
```

```
contador=besselj(0,2*%pi*To*fD);
i=2;

while abs(contador)<0.1 & i<K
    contador=contador+besselj(0,2*%pi*i*To*fD);
    i=i+1;
end

//Nopt=i;

//cálculo de Nopt con el método 2

Nopt=5;
//Nopt=50;


M=floor(K/Nopt);
J=0;
Jant=0;

//cálculo de h

h=zeros(1,K)+%i*zeros(1,K);

for k=0:M-1
    x=rand(1,["uniform"]);
    for ki=1:Nopt
        h(1,ki+k*Nopt)=x+%i*x;
    end
end

//cálculo de s retrasada

sr=zeros(1,K);

for n=1:1:K-retraso
    sr(n)=s(n+retraso);
```

```
end

for n=K-retraso+1:1:K
   sr(n)=0;
end

//cálculo del vector recibido r

r=a*(sr.*h)+v;

//cálculo del retraso

for ret=0:50
   sret=zeros(1,K);
   for n=1:1:K-ret
      sret(n)=s(n+ret);
   end
   for n=K-ret+1:1:K
      sret(n)=0;
   end

   for m=1:1:M
      for n=(m-1)*Nopt+1:1:m*Nopt
         J=J+r(:,n).*conj(sret(n));
      end
   end
   if norm(J)>norm(Jant)
      Jant=J;
      t=ret;
   end
   J=0;
end


endfunction
```

## 4. Código en Scilab de la función AOA con modificación en la red y en el terminal (estima del ángulo de llegada).

```
function ang=AOA

//datos iniciales del problema

K=50;
retraso=3;
Na=5;
alpha=60; //dato que buscamos

//cálculo de s,v, h y a

s=5*(rand(1,K,"normal")+%i*rand(1,K,"normal"));
//s=10*(rand(1,K,"normal")+%i*rand(1,K,"normal"));
//s=20*(rand(1,K,"normal")+%i*rand(1,K,"normal"));

v=rand(Na,K,"normal")+%i*rand(Na,K,"normal");
//h=ones(1,K);
//v=zeros(Na,K)+j*zeros(Na,K);



a=1;
for u=1:1:Na-1

a=[a;cos(2*%pi*(0.5)*sin(alpha*(%pi/180))*u)+%i*sin(2*%pi*(0.5)*sin(alpha*(%pi/180))*
u)];
end

//cálculo de s(n) retardada

sr=zeros(1,K);

for n=1:1:K-retraso
   sr(1,n)=s(1,n+retraso);
end
```

```
for n=K-retraso+1:1:K
   sr(1,n)=0;
end

h=rand(1,K,"uniform");

//esto serviría para calcular una secuencia h que variara sólo en cada intervalo

h=rand(1,K,"uniform");
//h=zeros(1,K)+%i*zeros(1,K);

//for k=0:M-1
//    x=rand(1,"uniform");
//    for ki=1:N
//        h(1,ki+k*5)=x+%i*x;
//    end
//end


r=a*(sr.*h)+v;

//cálculo de z y p

z=zeros(K,Na);
p=zeros(K,1);


for m=1:K
   z(m,:)=(r(:,m).*conj(sr(1,m)))';
end


for m=1:K
   p(m)=abs(sr(m))^2;
end
```

```
//estima de h

for d=1:1:K
    if p(d)~=0
        hest(d)=z(d,1)/p(d);
    else
        hest(d)=0;
    end
end

//cálculo de la matriz A

A=p(1)*hest(1)*eye(Na,Na);
//A=p(1)*eye(Na,Na);

zf=z(1,:);

for b=2:K
    zf=[zf,z(b,:)];
end

zf=zf';

for c=2:K
   A=[A;p(c)*hest(c)*eye(Na,Na)];
   //A=[A;p(c)*eye(Na,Na)];
end

//cálculo del ángulo

aest=inv((A')*A)*(A')*zf;

ang=abs((asin((acos(real(aest(2))))/%pi)/%i)*180/%pi);

while ang>360
    ang=ang-360;
```

```
end

while ang<0
   ang=ang+360;
end

endfunction
```