Apéndices

Apéndice **A**

Códigos programados para Matlab.

En esta sección se muestran los códigos de los algoritmos de la FFT programados en Matlab y los códigos de las pruebas para la evaluación de los algoritmos.

A.1. Algoritmos de la FFT.

A.1.1. Radix-2 DIT.

```
    % Algoritmo Radix-2 DIT
function [X] = fftradix2dit(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=bitrevorder(x); % Decimación de la señal.
for e=1:v, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
    r=2^(v-e);
    W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
    n2=1;
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        for n=0:(L/2)-1,
            a1=n+k+1;
            b1=(L/2)+a1;
            a=X(a1); % Carga valor 1.
            b=X(b1)*W(n2); % Carga valor 2 * factor de fase.
            X(a1) = a + b; % Salida 1 de la mariposa.
            X(b1) = a - b; % Salida 2 de la mariposa.
            n2=n2+1; % Proceso para el siguiente factor de fase.
            if n2>n1+1,
                n2=1;
            end
        end
    end % Fin de mariposas.
end % Fin de etapa.
end % Fin de función.
```

A.1.2. Radix-2 DIF.

```
    % Algoritmo Radix-2 DIF
function [X] = fftradix2dif(x)
```

```
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=x; % Reasignación de la señal original.
for e=v:-1:1, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
  r=2^(v-e);
  W=exp((-2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
  n2=1;
  for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
      for n=0:(L/2)-1,
          a1=n+k+1;
          b1=(L/2)+a1;
          a=X(a1); % Carga valor 1.
          b=X(b1); % Carga valor 2.
          X(a1) = a + b; % Salida 1 de la mariposa.
          X(b1) = (a - b) * W(n2); % Salida 2 de la mariposa.
          n2=n2+1; % Proceso para el sig. factor de fase.
          if n2>n1+1,
              n2=1;
          end
      end
  end % Fin de mariposas.
end % Fin de etapa.
X=bitrevorder(x); % Reacomodo de la señal de salida.
end % Fin de función.
```

A.1.3. Radix-4 DIT.

```
    %Algoritmo Radix-4 DIT
function [X] = fftradix4dit(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x; % Reasignación de la entrada.
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
for e=1:v, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b)*W((1*n1*n2)+1); % Carga valor 2 * Factor de fase.
            C=X(c)*W((2*n1*n2)+1); % Carga valor 3* Factor de fase.
            D=X(d)*W((3*n1*n2)+1); % Carga valor 4* Factor de fase.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D)); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D)); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
end % Fin de función.
```

A.1.4. Radix-4 DIF.

```
 %Algoritmo Radix-4 DIF
function [X] = fftradix4dif(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x;
W=exp((-2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
for e=v:-1:1, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b); % Carga valor 2.
            C=X(c); % Carga valor 3.
            D=X(d); % Carga valor 4.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D))*W((1*n1*n2)+1); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D)*W((2*n1*n2)+1); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D))*W((3*n1*n2)+1); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
end % Fin de función.
```

A.2. Algoritmos inversos de la FFT.

A.2.1. Radix-2 DIT inverso.

```
    % Algoritmo Radix-2 DIT inverso
function [X] = fftradix2dit(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=bitrevorder(x); % Decimación de la señal.
for e=1:v, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
    r=2^(v-e);
    W=exp((2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
    n2=1;
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        for n=0:(L/2)-1,
            a1=n+k+1;
            b1=(L/2)+a1;
            a=X(a1); % Carga valor 1.
            b=X(b1)*W(n2); % Carga valor 2 * factor de fase.
            X(a1) = a + b; % Salida 1 de la mariposa.
            X(b1) = a - b; % Salida 2 de la mariposa.
            n2=n2+1; % Proceso para el siguiente factor de fase.
            if n2>n1+1,
                n2=1;
```

```
            end
        end
    end % Fin de mariposas.
end % Fin de etapa.
X=X/N;
end % Fin de función.
```

A.2.2. Radix-2 DIF inverso.

```
    % Algoritmo Radix-2 DIF inverso
function [X] = fftradix2dif(x)
N=length(x); % Long. de la señal de entrada.
v=log2(N); % Núm. de etapas.
N2=N/2;
X=x; % Reasignación de la señal original.
for e=v:-1:1, % Inicio de etapa.
  L=2^e; % Condiciones iniciales.
  n1=(2^(e-1))-1;
  r=2^(v-e);
  W=exp((2*pi*i)*[0:n1]*r/N); % Cálculo de factores de fase.
  n2=1;
  for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
      for n=0:(L/2)-1,
          a1=n+k+1;
          b1=(L/2)+a1;
          a=X(a1); % Carga valor 1.
          b=X(b1); % Carga valor 2.
          X(a1) = a + b; % Salida 1 de la mariposa.
          X(b1) = (a - b) * W(n2); % Salida 2 de la mariposa.
          n2=n2+1; % Proceso para el sig. factor de fase.
          if n2>n1+1,
              n2=1;
          end
      end
  end % Fin de mariposas.
end % Fin de etapa.
X=bitrevorder(x)/N; % Reacomodo de la señal de salida.
end % Fin de función.
```

A.2.3. Radix-4 DIT inverso.

```
    %Algoritmo Radix-4 DIT inverso
function [X] = fftradix4dit(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x; % Reasignación de la entrada.
W=exp((2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
X=digitrevorder(X,4); % Reacomodo de la señal de salida.
for e=1:v, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b)*W((1*n1*n2)+1); % Carga valor 2 * Factor de fase.
```

```
            C=X(c)*W((2*n1*n2)+1); % Carga valor 3* Factor de fase.
            D=X(d)*W((3*n1*n2)+1); % Carga valor 4* Factor de fase.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D)); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D)); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=X/N;
end % Fin de función.
```

A.2.4. Radix-4 DIF inverso.

```
 %Algoritmo Radix-4 DIF inverso
function [X] = fftradix4dif(x)
N=length(x); % Long. de la señal de entrada.
v=log(N)/log(4); % Núm. de etapas.
N4=N/4;
X=x;
W=exp((2*pi*i)*[0:N-1]/N); % Cálculo de los factores de fase.
for e=v:-1:1, % Inicio de etapa.
    L=4^e; % Condiciones iniciales.
    n2=4^(v-e);
    if e==1, n2=0; end
    for k=0:L:(N-L), % Inicio del cálculo de las mariposas.
        n1=0;
        for n=0:(L/4)-1,
            a=n+k+1;
            b=(L/4)+a;
            c=(L/4)+b;
            d=(L/4)+c;
            A=X(a); % Carga valor 1.
            B=X(b); % Carga valor 2.
            C=X(c); % Carga valor 3.
            D=X(d); % Carga valor 4.
            X(a)=A+B+C+D; % Salida 1 de la mariposa.
            X(b)=(A-(j*B)-C+(j*D))*W((1*n1*n2)+1); % Salida 2 de la mariposa.
            X(c)=(A-B+C-D)*W((2*n1*n2)+1); % Salida 3 de la mariposa.
            X(d)=(A+(j*B)-C-(j*D))*W((3*n1*n2)+1); % Salida 4 de la mariposa.
            n1=n1+1; % Sig. factor de fase
        end
    end % Fin de mariposa.
end % Fin de etapa.
X=digitrevorder(X,4)/N; % Reacomodo de la señal de salida.
end % Fin de función.
```

A.3. Pruebas a los algoritmos.

A.3.1. Comparación con FFT de Matlab.

```
  %Pruebas para los algoritmos Radix
load yt; %Carga señal 1
load yes; %Carga canales de comunicación

alg='Radix-2 DIT';
YT=fftradix2dit(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}'); %Llama programa de graficación 2
```

```
title(alg)
YT1=fftradix2dit(yc1);
YT2=fftradix2dit(yc2);
YT3=fftradix2dit(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3); %Llama programa de graficación 3

alg='Radix-2 DIF';
YT=fftradix2dif(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}');
title(alg)
YT1=fftradix2dif(yc1);
YT2=fftradix2dif(yc2);
YT3=fftradix2dif(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3);

alg='Radix-4 DIT';
YT=fftradix4dit(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}');
title(alg)
YT1=fftradix4dit(yc1);
YT2=fftradix4dit(yc2);
YT3=fftradix4dit(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3);

alg='Radix-4 DIF';
YT=fftradix4dif(yt);
figure
subplot(2,1,1), graf2(yt,YT,'H_{yt}');
title(alg)
YT1=fftradix4dif(yc1);
YT2=fftradix4dif(yc2);
YT3=fftradix4dif(yc3);
subplot(2,1,2), graf3(yc1,yc2,yc3,YT1,YT2,YT3);

  %Graficación 2
function graf2(yt,YT,alg)
%alg = nombre del algoritmo
N2 = length(yt)/2;
a=pi/N2;
t=0:a:pi-a;
plot(t,abs(YT(1:N2)),'g')
hold on
YTM=abs(fft(yt));
plot(t,YTM(1:N2),'b-.')
legend(alg,'FFT Matlab','Location','Best')
ylabel('Amplitud')
xlabel('Frecuencia [Rad]')
end

  %Graficación 3
function graf3(y1,y2,y3,YT1,YT2,YT3)
%alg = nombre del algoritmo
N2 = length(y1)/2;
a=pi/N2;
t=0:a:pi-a;
plot(t,abs(YT1(1:N2)),'g')
hold on
plot(t,abs(YT2(1:N2)),'r')
plot(t,abs(YT3(1:N2)),'y')
YTM1=abs(fft(y1));
plot(t,YTM1(1:N2),'b-.')
YTM2=abs(fft(y2));
plot(t,YTM2(1:N2),'b-.')
```

```
YTM3=abs(fft(y3));
plot(t,YTM3(1:N2),'b-.')
legend('H_{c1}','H_{c2}','H_{c3}','Matlab','Location','Best')
ylabel('Amplitud')
xlabel('Frecuencia [Rad]')
end
```

A.3.2. Precisión del módulo.

```
  %Precisión Módulo
N=N+1;
yt=randn(1,2^N); % Generando la señal de entrada.
v=fft(yt); % Cálculo de FFT Matlab.

%Radix-2 DIT
YT=fftradix2dit(yt); % Cálculo del algoritmo.
iv=ifftradix2dit(YT); % Cálculo del algoritmo inverso.
efr2t=abs(v)-abs(YT); % Diferencia hacia adelante.
ebr2t=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer2tf=0;
eer2tb=0;
for n=1:2^N,
    eer2tf=(efr2t(n)^2) + eer2tf; % Cálculo error cuadrático promedio hacia adelante.
    eer2tb=(ebr2t(n)^2)+ eer2tb; % Cálculo error cuadrático promedio hacia atrás.
end
epcr2tf(N)=(eer2tf/2^N);
epcr2tb(N)=(eer2tb/2^N);
dsfr2tp=0;
dsbr2tp=0;
for n=1:2^N,
    dsfr2tp=(efr2t(n)-epcr2tf(N))^2 + dsfr2tp; % Cálculo varianza adelante.
    dsbr2tp=(ebr2t(n)-epcr2tb(N))^2 + dsbr2tp; % Cálculo varianza atrás.
end
dsfr2t(N)=dsfr2tp/2^N;
dsbr2t(N)=dsbr2tp/2^N;

%Radix-2 DIF
YT=fftradix2dif(yt); % Cálculo del algoritmo.
iv=ifftradix2dif(YT); % Cálculo del algoritmo inverso.
efr2f=abs(v)-abs(YT);  % Diferencia hacia adelante.
ebr2f=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer2ff=0;
eer2fb=0;
for n=1:2^N,
    eer2ff=(efr2f(n)^2) + eer2ff; % Cálculo error cuadrático promedio adelante.
    eer2fb=(ebr2f(n)^2)+ eer2fb; % Cálculo error cuadrático promedio atrás.
end
epcr2ff(N)=(eer2ff/2^N);
epcr2fb(N)=(eer2fb/2^N);
dsfr2fp=0;
dsbr2fp=0;
for n=1:2^N,
    dsfr2fp=(efr2f(n)-epcr2ff(N))^2 + dsfr2fp; % Cálculo varianza adelante.
    dsbr2fp=(ebr2f(n)-epcr2fb(N))^2 + dsbr2fp; % Cálculo varianza atrás.
end
dsfr2f(N)=dsfr2fp/2^N;
dsbr2f(N)=dsbr2fp/2^N;

if N<=6,
yt=randn(1,4^N);
v=fft(yt);

%Radix-4 DIT
YT=fftradix4dit(yt); % Cálculo del algoritmo.
```

```
iv=ifftradix4dit(YT); % Cálculo del algoritmo inverso.
efr4t=abs(v)-abs(YT);  % Diferencia hacia adelante.
ebr4t=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer4tf=0;
eer4tb=0;
for n=1:4^N,
    eer4tf=(efr4t(n)^2) + eer4tf; % Cálculo error cuadrático promedio adelante.
    eer4tb=(ebr4t(n)^2) + eer4tb; % Cálculo error cuadrático promedio atrás.
end
epcr4tf(N)=(eer4tf/4^N); % Cálculo varianza hacia adelante.
epcr4tb(N)=(eer4tb/4^N); % Cálculo varianza hacia atrás.
dsfr4tp=0;
dsbr4tp=0;
for n=1:4^N,
    dsfr4tp=(efr4t(n)-epcr4tf(N))^2 + dsfr4tp;
    dsbr4tp=(ebr4t(n)-epcr4tb(N))^2 + dsbr4tp;
end
dsfr4t(N)=dsfr4tp/4^N;
dsbr4t(N)=dsbr4tp/4^N;

%Radix-4 DIF
YT=fftradix4dif(yt); % Cálculo del algoritmo.
iv=ifftradix4dif(YT); % Cálculo del algoritmo inverso.
efr4f=abs(v)-abs(YT);  % Diferencia hacia adelante.
ebr4f=abs(yt)-abs(iv); % Diferencia hacia atrás.
eer4ff=0;
eer4fb=0;
for n=1:4^N,
    eer4ff=(efr4f(n)^2) + eer4ff; % Cálculo error cuadrático promedio adelante.
    eer4fb=(ebr4f(n)^2) + eer4fb; % Cálculo error cuadrático promedio atrás.
end
epcr4ff(N)=(eer4ff/4^N);
epcr4fb(N)=(eer4fb/4^N);
dsfr4fp=0;
dsbr4fp=0;
for n=1:4^N,
    dsfr4fp=(efr4f(n)-epcr4ff(N))^2 + dsfr4fp; % Cálculo varianza adelante.
    dsbr4fp=(ebr4f(n)-epcr4fb(N))^2 + dsbr4fp; % Cálculo varianza atrás.
end
dsfr4f(N)=dsfr4fp/4^N;
dsbr4f(N)=dsbr4fp/4^N;
end

if N~=12, % Siguiente longitud de señal de entrada.
    precimodulo;
end

if N==12, % Graficación
    for n=1:12, t(n)=2^n; end
    for n=1:6, t1(n)=4^n; end

    figure
    loglog(t,epcr2tf,'g-+')
    hold on
    loglog(t,epcr2ff,'r-o')
    loglog(t1,epcr4tf,'b-*')
    loglog(t1,epcr4ff,'m-x')
    legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
    title('Error Promedio Cuadrático hacia adelante en el Módulo')
    xlabel('Número de puntos')
    ylabel('Error')

    figure
    loglog(t,epcr2tb,'g-.+')
    hold on
```

```
    loglog(t,epcr2fb,'r-.o')
    loglog(t1,epcr4tb,'b-.*')
    loglog(t1,epcr4fb,'m-.x')
    legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
    title('Error Promedio Cuadrático hacia atrás en el Módulo')
    xlabel('Número de puntos')
    ylabel('Error')
```

A.3.3. Precisión del ángulo de fase.

```
  %Precisión Ángulo de fase
N=N+1;
yt=randn(1,2^N); % Generando la señal de entrada.
v=fft(yt); % Cálculo de FFT Matlab.

%Radix-2 DIT
YT=fftradix2dit(yt); % Cálculo del algoritmo.
iv=ifftradix2dit(YT); % Cálculo del algoritmo inverso.
efr2t=angle(v)-angle(YT); % Diferencia hacia adelante.
ebr2t=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer2tf=0;
eer2tb=0;
for n=1:2^N,
    eer2tf=(efr2t(n)^2) + eer2tf; % Cálculo error cuadrático promedio hacia adelante.
    eer2tb=(ebr2t(n)^2)+ eer2tb; % Cálculo error cuadrático promedio hacia atrás.
end
epcr2tf(N)=(eer2tf/2^N);
epcr2tb(N)=(eer2tb/2^N);
dsfr2tp=0;
dsbr2tp=0;
for n=1:2^N,
    dsfr2tp=(efr2t(n)-epcr2tf(N))^2 + dsfr2tp; % Cálculo varianza adelante.
    dsbr2tp=(ebr2t(n)-epcr2tb(N))^2 + dsbr2tp; % Cálculo varianza atrás.
end
dsfr2t(N)=dsfr2tp/2^N;
dsbr2t(N)=dsbr2tp/2^N;

%Radix-2 DIF
YT=fftradix2dif(yt); % Cálculo del algoritmo.
iv=ifftradix2dif(YT); % Cálculo del algoritmo inverso.
efr2f=angle(v)-angle(YT);  % Diferencia hacia adelante.
ebr2f=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer2ff=0;
eer2fb=0;
for n=1:2^N,
    eer2ff=(efr2f(n)^2) + eer2ff; % Cálculo error cuadrático promedio adelante.
    eer2fb=(ebr2f(n)^2)+ eer2fb; % Cálculo error cuadrático promedio atrás.
end
epcr2ff(N)=(eer2ff/2^N);
epcr2fb(N)=(eer2fb/2^N);
dsfr2fp=0;
dsbr2fp=0;
for n=1:2^N,
    dsfr2fp=(efr2f(n)-epcr2ff(N))^2 + dsfr2fp; % Cálculo varianza adelante.
    dsbr2fp=(ebr2f(n)-epcr2fb(N))^2 + dsbr2fp; % Cálculo varianza atrás.
end
dsfr2f(N)=dsfr2fp/2^N;
dsbr2f(N)=dsbr2fp/2^N;

if N<=6,
yt=randn(1,4^N);
v=fft(yt);

%Radix-4 DIT
```

```
YT=fftradix4dit(yt); % Cálculo del algoritmo.
iv=ifftradix4dit(YT); % Cálculo del algoritmo inverso.
efr4t=angle(v)-angle(YT);  % Diferencia hacia adelante.
ebr4t=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer4tf=0;
eer4tb=0;
for n=1:4^N,
    eer4tf=(efr4t(n)^2) + eer4tf; % Cálculo error cuadrático promedio adelante.
    eer4tb=(ebr4t(n)^2) + eer4tb; % Cálculo error cuadrático promedio atrás.
end
epcr4tf(N)=(eer4tf/4^N); % Cálculo varianza hacia adelante.
epcr4tb(N)=(eer4tb/4^N); % Cálculo varianza hacia atrás.
dsfr4tp=0;
dsbr4tp=0;
for n=1:4^N,
    dsfr4tp=(efr4t(n)-epcr4tf(N))^2 + dsfr4tp;
    dsbr4tp=(ebr4t(n)-epcr4tb(N))^2 + dsbr4tp;
end
dsfr4t(N)=dsfr4tp/4^N;
dsbr4t(N)=dsbr4tp/4^N;

%Radix-4 DIF
YT=fftradix4dif(yt); % Cálculo del algoritmo.
iv=ifftradix4dif(YT); % Cálculo del algoritmo inverso.
efr4f=angle(v)-angle(YT);  % Diferencia hacia adelante.
ebr4f=angle(yt)-angle(iv); % Diferencia hacia atrás.
eer4ff=0;
eer4fb=0;
for n=1:4^N,
    eer4ff=(efr4f(n)^2) + eer4ff; % Cálculo error cuadrático promedio adelante.
    eer4fb=(ebr4f(n)^2) + eer4fb; % Cálculo error cuadrático promedio atrás.
end
epcr4ff(N)=(eer4ff/4^N);
epcr4fb(N)=(eer4fb/4^N);
dsfr4fp=0;
dsbr4fp=0;
for n=1:4^N,
    dsfr4fp=(efr4f(n)-epcr4ff(N))^2 + dsfr4fp; % Cálculo varianza adelante.
    dsbr4fp=(ebr4f(n)-epcr4fb(N))^2 + dsbr4fp; % Cálculo varianza atrás.
end
dsfr4f(N)=dsfr4fp/4^N;
dsbr4f(N)=dsbr4fp/4^N;
end

if N~=12, % Siguiente longitud de señal de entrada.
    precifase;
end

if N==12, % Graficación
    for n=1:12, t(n)=2^n; end
    for n=1:6, t1(n)=4^n; end

    figure
    loglog(t,fepcr2tf,'g-+')
    hold on
    loglog(t,fepcr2ff,'r-o')
    loglog(t1,fepcr4tf,'b-*')
    loglog(t1,fepcr4ff,'m-x')
    legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
    title('Error Promedio Cuadrático hacia adelante en la fase')
    xlabel('Número de puntos')
    ylabel('Error')

    figure
    loglog(t,fepcr2tb,'g-.+')
```

```
hold on
loglog(t,fepcr2fb,'r-.o')
loglog(t1,fepcr4tb,'b-.*')
loglog(t1,fepcr4fb,'m-.x')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
title('Error Promedio Cuadrático hacia atrás en la fase')
xlabel('Número de puntos')
ylabel('Error')
```

A.3.4. Tiempo de cálculo en segundos.

```
    %Tiempo de cálculo
N=N+1;
yt=randn(1,2^N); % Generando la señal de entrada
%disp('Radix-2 DIT')
tic, % Inicia el cronómetro. R2-DIT
for k=1:10, % Número de iteraciones.
    YT=fftradix2dit(yt); % Función programada.
end % Fin de iteraciones.
tr2dit(N)=toc/10; % Para el cronómetro.
%err(yt,YT)

%disp('Radix-2 DIF')
tic, % Inicia el cronómetro. R2-DIF
for k=1:10, % Número de iteraciones.
    YT=fftradix2dif(yt); % Función programada.
end % Fin de iteraciones.
tr2dif(N)=toc/10; % Para el cronómetro.
%err(yt,YT)

%disp('FFT Matlab')
tic, % Inicia el cronómetro. FFT-Matlab.
for k=1:10, % Número de iteraciones.
    YT=fft(yt); % Función programada.
end % Fin de iteraciones.
tfft(N)=toc/10; % Para el cronómetro.

if N<=6,
yt=randn(1,4^N);
% disp('Radix-4 DIT')
tic, % Inicia el cronómetro. R4-DIT
for k=1:10, % Número de iteraciones.
    YT=fftradix4dit(yt); % Función programada.
end % Fin de iteraciones
tr4dit(N)=toc/10; % Para el cronómetro.
%err(yt,YT)

% disp('Radix-4 DIF')
tic, % Inicia el cronómetro. R4-DIF
for k=1:10, % Número de iteraciones.
    YT=fftradix4dif(yt); % Función programada.
end % Fin de iteraciones
tr4dif(N)=toc/10; % Para el cronómetro.
%err(yt,YT)
end
if N~=12,
    tical;
end

if N==12, % Graficación
figure
loglog(t,tr2dit,'g+-')
hold on
loglog(t,tr2dif,'ro-')
```

```
loglog(t1,tr4dit,'b*-')
loglog(t1,tr4dif,'mx-')
loglog(t,tfft,'kv-')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Matlab','Location','Best')
title('Tiempo de cálculo en segundos')
xlabel('Número de puntos')
ylabel('Segundos')
```

A.3.5. Tiempo de cálculo en "mflops´´.

```
  %mflops
for N=1:12, % Cálculo de mflops para Radix-2
    mfr2f(N)=((5*(2^N)*log2(2^N))./(tr2dif(N)*10^6));
    mfr2t(N)=((5*(2^N)*log2(2^N))./(tr2dit(N)*10^6));
    mffft(N)=((5*(2^N)*log2(2^N))./(tfft(N)*10^6));
end
for N=1:6, % Cálculo de mflops para Radix-4
    mfr4t(N)=((5*(4^N)*log2(4^N))./(tr4dit(N)*10^6));
    mfr4f(N)=((5*(4^N)*log2(4^N))./(tr4dif(N)*10^6));
end

figure % Graficiación
loglog(t,mfr2t,'g+-')
hold on
loglog(t,mfr2f,'ro-')
loglog(t1,mfr4t,'b*-')
loglog(t1,mfr4f,'mx-')
loglog(t,mffft,'kv-')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Matlab','Location','Best')
title('Velocidad de cálculo')
xlabel('Número de puntos')
ylabel('mFlops')

figure
semilogx(t,mfr2t,'g+-')
hold on
semilogx(t,mfr2f,'ro-')
semilogx(t1,mfr4t,'b*-')
semilogx(t1,mfr4f,'mx-')
%semilogx(t,mffft,'ko-')
legend('R-2 DIT','R-2 DIF','R-4 DIT','R-4 DIF','Location','Best')
title('Velocidad de cálculo')
xlabel('Número de puntos')
ylabel('mFlops')
```