Apéndice.

Códigos.

Para la realización de este proyecto se han empleado funciones internas de Matlab, así como funciones desarrolladas por el Grupo de Investigación "Física Interdisciplinar. Fundamentos y Aplicaciones" dirigido por el Prof. Dr. Emilio Gómez González en la Escuela Superior de Ingenieros de la Universidad de Sevilla. Algunas de estas funciones utilizadas han sido desarrolladas específicamente para la elaboración de este proyecto.

A continuación, mostramos la relación de funciones desarrolladas específicamente para la elaboración de este proyecto, así como sus códigos:

- calcdesp.m
- calcmono.m
- calc3D.m
- pincho.m
- opcfinal.m
- vermono.m
- ver3D.m

```
%                       calcdesp.m

%           Programa realizado por Javier González Cuenca


function varargout=calcdesp(varargin)

% Este es el cuerpo principal del programa de cálculo de los
% desplazamientos.
% A partir de aquí, iremos preguntando todos los parámetros de los que
% depende el problema.
% A esta función se le puede llamar de varias formas.
% Si se la llama con un argumento de retorno el programa calcula el
campo
% monodimensional, preguntando los datos que le hacen falta para ello.
% Si se la llama con tres argumentos de retorno, le campo que calcula
es
% tridimensional.
% Si no tiene argumentos de retorno el programa preguntará como es el
% campo a calcular, monodimensional o tridimensional.


close all;
clear all;
clc;


global d_OBJ_CCD LONG_ONDA PIXEL E0_ONDA TIPAPROX


Dz=[]; % Aquí almacenaremos los desplazamientos en la dirección z de
       cada
        % píxel.
Dx=[]; % dirección x
Dy=[]; % dirección y
```

```
tqr=find(nargout==[0 1 3]);      % dependiendo del número de
argumentos
                                 % de retorno ejecuta diferentes
rutinas.
if((isempty(tqr))==1)
   errordlg('Número de argumentos de salida no válido')
   return
end
switch(nargout)
case 1
   varargout(1)={Dz};
case 3
   varargout(1)={Dx};
   varargout(2)={Dy};
   varargout(3)={Dz};
end


h1=helpdlg('Bienvenidos al programa de cálculo de desplazamientos
utilizando técnicas de interferometría holográfica digital. A
continuación le irán apareciendo las distintas opciones de trabajo que
pueden llevarse a cabo.');
waitfor(h1);


% A continuación se pregunta por el tipo de holograma. De momento, el
% caso de hologramas de
% transmisión no ha sido desarrollado, debido a las pocas aplicaciones
% mecánicas de los
% mismos ( casi todos los sólidos son opacos ).

tiphol1=menu('TIPO DE  HOLOGRAMA','Reflexión','Transmisión');
while(isempty(tiphol1))
   h2=warndlg('Seleccione una opción válida');
   wairfor(h2)
   tiphol1=menu('TIPO DE  HOLOGRAMA','Reflexión','Transmisión');
end
tiphol=tiphol1-1;
```

```
if(tiphol==1)
   errordlg('Hologramas de transmisión no disponibles')
   return
end



% Aquí pregunto sí el objeto es cuasiplano para ver que tipo de
% reconstrucción voy a tener
% que hacer.Si es cuasi plano utilizaré la aprximación FFT para la
% reconstrucción del
% campo electromagnético, ya que parto de la base que los haces de
% iluminación son
% colimados(si no lo son el programa se corta mas adelante).
% Si el objeto de caracter 3D, resolveremos la integral
% de Fresnel-Kirchoff

ap1=menu('Su objeto es cuasi plano','Sí','No');

while(isempty(ap1))
   h5=warndlg('Seleccione una opción válida');
   waitfor(h5)
   ap1=menu('Su objeto es cuasi plano','Sí','No');
end

ap=ap1-1;

TIPAPROX=ap;



% Preguntamos al usuario por la energía característica del laser
empleado
% en el experimento

def={'30'};
lineNo=1;
AddOpts.Resize='on';
```

```
AddOpts.WindowStyle='normal';
AddOpts.Interpreter='tex';

E1=inputdlg('Introduzca el valor de la amplitud del campo eléctrico E0
según características del láser (mW):');

if(isempty(E1))
   return
end

ay=eval(E1{1},'-1');

while(ay==-1)
   h3=warndlg('E0 debe ser una cantidad real positiva.');
   waitfor(h3)
   E1=inputdlg('Introduzca el valor de la amplitud del campo eléctrico
E0 según características del láser (mW):');
   if(isempty(E1))
      return
   end
   ay=eval(E1{1},'-1');
end

E=eval(E1{1});

while(E<=0)
   h3=warndlg('E0 debe ser una cantidad real positiva.');
   waitfor(h3)
   E1=inputdlg('Introduzca el valor de la amplitud del campo eléctrico
E0 según características del láser (mW):');
   if(isempty(E1))
      return
   end
   ay=eval(E1{1},'-1');
   while(ay==-1)
      h3=warndlg('E0 debe ser una cantidad real positiva.');
      waitfor(h3)
```

```
      E1=inputdlg('Introduzca el valor de la amplitud del campo
eléctrico E0 según características del láser (mW):');
      if(isempty(E1))
         return
      end
      ay=eval(E1{1},'-1');
   end
   E=eval(E1{1});
end

E0_ONDA=E/1000;




% Preguntamos al usuario por la longitud de onda del laser empleado en
el
% experimento.

Londa1=inputdlg('Introduzca la Longitud de Onda del haz incidente
(nm): ');

if(isempty(Londa1))
   return
end

ay=eval(Londa1{1},'-1');

while(ay==-1)
   h3=warndlg('La Longitud de Onda debe ser una cantidad real
positiva.');
   waitfor(h3)
   Londa1=inputdlg('Introduzca la Longitud de Onda del haz incidente
(nm): ');
   if(isempty(Londa1))
      return
   end
   ay=eval(Londa1{1},'-1');
```

```
end

Londa=eval(Londa1{1});

while(Londa<=0)
   h3=warndlg('La Longitud de Onda debe ser una cantidad real
positiva.');
   waitfor(h3)
   Londa1=inputdlg('Introduzca la Longitud de Onda del haz incidente
(nm): ');
   if(isempty(Londa1))
      return
   end
   ay=eval(Londa1{1},'-1');
   while(ay==-1)
      h3=warndlg('La Longitud de Onda debe ser una cantidad real
positiva.');
      waitfor(h3)
      Londa1=inputdlg('Introduzca la Longitud de Onda del haz
incidente (nm): ');
      if(isempty(Londa1))
         return
      end
      ay=eval(Londa1{1},'-1');
   end
   Londa=eval(Londa1{1});
end

LONG_ONDA=Londa/1000000000;




% Preguntamos al usuario por el tamaño de muestreo (tamaño del pixel
en
% el objeto).
% El lado del pixel es algo complicado de saber.Este problema no puede
% ser resuelto en
```

```
% este programa, ya que depende de las condiciones en las que se haya
% hecho el experimento.
% Proponemos como solución sencilla en el laboratorio, situar el
objeto
% dentro de un marco
% de dimensiones conocidas, y ajustar el tamaño de la imagen del CCD
al
% del marco,
% y asi calcular el tamaño de pixel.

dx1=inputdlg('Muestreo del objeto (mm/px): ');

if(isempty(dx1))
   return
end

ay=eval(dx1{1},'-1');

while(ay==-1)
   h3=warndlg('Lado del pixel debe de ser una cantidad real
positiva.');
   waitfor(h3)
   dx1=inputdlg('Muestreo del objeto (mm/px): ');
   if(isempty(dx1))
      return
   end
   ay=eval(dx1{1},'-1');
end

dx=eval(dx1{1});

while(dx<=0)
   h3=warndlg('Lado del pixel debe de ser una cantidad real
positiva.');
   waitfor(h3)
   dx1=inputdlg('Muestreo del objeto (mm/px): ');
   if(isempty(dx1))
      return
```

```
   end
   ay=eval(dx1{1},'-1');
   while(ay==-1)
      h3=warndlg('Lado del pixel debe de ser una cantidad real
positiva.');
      waitfor(h3)
      dx1=inputdlg('Muestreo del objeto (mm/px): ');
      if(isempty(dx1))
         return
      end
      ay=eval(dx1{1},'-1');
   end
   dx=eval(dx1{1});
end

PIXEL=dx/1000;




% Ahora pregunto al usuario por la distancia del ccd al objeto

zeta1=inputdlg('Introduzca el valor de la distancia que separa el
C.C.D. y el objeto (m):');

if(isempty(zeta1))
   return
end

ay=eval(zeta1{1},'-1');

while(ay==-1)
   h3=warndlg('La distacia debe ser una cantidad real positiva.');
   waitfor(h3)
   zeta1=inputdlg('Introduzca el valor de la distancia que separa el
C.C.D. y el objeto (m):');
   if(isempty(zeta1))
      return
```

```
   end
   ay=eval(zeta1{1},'-1');
end

zeta=eval(zeta1{1});

while(zeta<=0)
   h3=warndlg('La distacia debe ser una cantidad real positiva.');
   waitfor(h3)
   zeta1=inputdlg('Introduzca el valor de la distancia que separa el
C.C.D. y el objeto (m):');
   if(isempty(zeta1))
      return
   end
   ay=eval(zeta1{1},'-1');
   while(ay==-1)
      h3=warndlg('La distacia debe ser una cantidad real positiva.');
      waitfor(h3)
      zeta1=inputdlg('Introduzca el valor de la distancia que separa
el C.C.D. y el objeto (m):');
      if(isempty(zeta1))
      return
      end
      ay=eval(zeta1{1},'-1');
   end
   zeta=eval(zeta1{1});
end

d_OBJ_CCD=zeta;

% Si estamos en aproximación FFT, la distancia de reconstrucción no
debe
% ser "zeta" ya que sino el tamaño de píxel no será el correcto y no
% podremos correlacionar  puntos del  objeto con píxeles del campo
% reconstruido.
% Por evitar esto vamos a modificar la distancia de reconstrucción y
así
% eliminar el fenómeno de "aberración". Tendremos que ajustar esto mas
```

```
% adelante(justo después de leer los hologramas) ya que es importante
% que el holograma no tenga aberración ni en x ni en y.




% Una vez preguntados todos los parámetros ópticos del problema
llamamos
% a distintas funciones, dependiendo del campo que vayamos a calcular.
% Contemplamos el caso de llamar a la función sin argumentos de
retorno.
% Para eso utilizaremos la variable nargout que crea Matlab
% automáticamente al llamar a la
% función. Si nargout==0, preguntamos por el tipo de campo a
calcular(monodimensional o 3D)

if(nargout==0)
   dimensdesp1=menu('Cálculo del campo de
desplazamientos','Monodimensional (eje z)','Tridimensional 3D');
   while(isempty(dimensdesp1))
      h2=warndlg('Seleccione una opción válida');
      wairfor(h2)
      dimensdesp1=menu('Cálculo del campo de
desplazamientos','Monodimensional (eje z)','Tridimensional 3D')
   end
   dimensdesp=dimensdesp1-1;
   if(dimensdesp==0)        % Estaríamos en el caso monodimensional.
      Dz=calcmono;   % Llamamos a la función que me calcula el campo
                     % monodimensional
   end
   if(dimensdesp==1)        % Estaríamos en el caso 3D.
      [Dx,Dy,Dz]=calc3D;    % Llamamos a la función que me calcula el
                            % campo 3D.
   end
end
```

```
% Si se llamó con un argumento de retorno, el campo es
monodimensional.

if(nargout==1)
   Dz=calcmono;             % Llamamos a la función que me calcula el
                            % campo monodimensional
   varargout(1)={Dz};       % Devolvemos Dz como argumento de retorno
end



% Si se llamó con tres argumentos de retorno, el campo es 3D.

if(nargout==3)
   [Dx,Dy,Dz]=calc3D;
   varargout(1)={Dx};       % Devolvemos como argumentos de retorno
los
                            % campos de desplazamientos
   varargout(2)={Dy};       % calculados Dx,Dy,Dz en ese orden
                            % sobre la primera,
   varargout(3)={Dz};       % segunda y tercera variable de llamada.
end
```

```
%                              calcmono.m

%                Programa realizado por Javier González Cuenca


function [Dz]=calcmono

% función que calcula el campo de desplazamientos monodimensional(eje
z).
% Para que los cálculos sean correctos, o el vector de sensibilidad es
% paralelo al eje z o el campo de desplazamientos dx,dy,dz tiene sólo
% componente z ( dx=0,dy=0 )
% No tiene argumentos de entrada, ya que necesita los valores antes
% preguntados al usuario que se pasan como variables globales.
% Esta función necesita conocer los archivos bmp donde se han
almacenados
% los dos hologramas del objeto (situación original y deformada), el
% desplazamiento y posición del punto de referencia (en el caso de
campo
% de desplazamientos absolutos) y la geometría del montaje (es decir,
los
% vectores de sensibilidad).


global d_OBJ_CCD LONG_ONDA PIXEL E0_ONDA TIPAPROX

Dz=[0];

% Se pregunta al usuario por el tipo de montaje utilizado.
% Si no es con vector de sensibilidad constante el programa se corta
% debido a la dificultad en la entrada de datos para todos los puntos.
% Proponemos como solución la entrada de datos mediante un fichero
.mat
% en el que este almacenada la componente zeta de los vectores de
% sensibilidad de cada pixel.

tipsens1=menu('¿Su montaje es con vector de sensibilidad
constante?','Sí','No');
while(isempty(tipsens1))
   h2=warndlg('Seleccione una opción válida');
   wairfor(h2)
   tipsens1=menu('¿Su montaje es con vector de sensibilidad
constante?','Sí','No');
end
tipsens=tipsens1-1;
if(tipsens==1)
   errordlg('No disponible para vector de sensibilidad variable')
   return
end



% Ahora pregunto al usuario por el vector dirección de iluminación.
pront={'Componente x:','Componente y:','Componente z:'};
def={'0','0','-1'};
lineNo=1;
```

```
vectorilu1=inputdlg(pront,'Vector dirección de
iluminación:',lineNo,def);
if(isempty(vectorilu1))
   return
end
while((isempty(vectorilu1{1}))|(isempty(vectorilu1{2}))|(isempty(vecto
rilu1{3})))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu1=inputdlg(pront,'Vector dirección de
iluminación:',lineNo,def);
   if(isempty(vectorilu1))
      return
   end
end
ay1=eval(vectorilu1{1},'-1*i');
ay2=eval(vectorilu1{2},'-1*i');
ay3=eval(vectorilu1{3},'-1*i');
while((ay1==-1*i)|(ay2==-1*i)|(ay3==-1*i))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu1=inputdlg(pront,'Vector dirección de
iluminación',lineNo,def);
   if(isempty(vectorilu1))
      return
   end

while((isempty(vectorilu1{1}))|(isempty(vectorilu1{2}))|(isempty(vecto
rilu1{3})))
      h3=warndlg('Vector no válido.');
      waitfor(h3)
      vectorilu1=inputdlg(pront,'Vector dirección de
iluminación',lineNo,def);
      if(isempty(vectorilu1))
         return
      end
   end
   ay1=eval(vectorilu1{1},'-1*i');
   ay2=eval(vectorilu1{2},'-1*i');
   ay3=eval(vectorilu1{3},'-1*i');

end

vectorilu=[eval(vectorilu1{1}) eval(vectorilu1{2})
eval(vectorilu1{3})];
     % Almaceno en este vector las componentes i,j,k del vector
     % iluminación en ejes de
     % globales del objeto.


moduloilu=sqrt(vectorilu(1)^2+vectorilu(2)^2+vectorilu(3)^2);
     % calculo el módulo del vector dirección de iluminación.

vectorilu=vectorilu/moduloilu;
     % normalizo la dirección de iluminación, por si acaso se ha
     % introducido sin normalizar.
```

```
vectorobs=[0 0 1];
     % el vector de observación siempre es el mismo, normal al CCD

vectsensi=(2*pi/LONG_ONDA)*(vectorobs-vectorilu);
     %calculo del vector de sensibilidad, como  [e]=2*pi/L_onda*([b]-
[s])



% Necesitamos saber si el campo de desplazamientos que vamos a
calcular
% es relativo o absoluto.

tipcamp1=menu('El campo de desplazamientos a calcular
es:','Relativo','Absoluto');

while(isempty(tipcamp1))
   h2=warndlg('Seleccione una opción válida');
   wairfor(h2)
   tipcamp1=menu('El campo de desplazamientos a calcular
es:','Relativo','Absoluto');
end

tipcamp=tipcamp1-1;

% Si el campo que queremos calcular es absoluto tenemos que introducir
al
% desplazamiento y coordenadas(en píxel, es decir, en el plano del
CCD)
% del punto de referencia



% Para el caso de campo de desplazamientos absoluto necesitamos
% introducir el punto de referencia y su desplazamiento.
% Si el campo es absoluto, necesito un punto de referencia. Este punto
de
% referencia del objeto concuerda con un píxel del CCD (según la
% dirección de observación).
% Podemos reconocer este punto mediante cambio de sistemas de
coordenadas
% (de objeto a CCD) o localizándolo en una foto del objeto (foto, no
% holograma) hecha con el CCD en la misma posición del experimento.

% Si elegimos la opción de introducir las coordenadas del píxel en el
% plano de reconstrucción que concuerda(según la dirección de
% observación) con el punto de  referencia, tenemos que realizar un
% cambio en coordenadas para localizar los píxeles.
% Este es un problema que debe ser previamente resuelto a la ejecución
% del programa, deducido a partir de las condiciones del experimento.

% Si por el contrario elegimos localizar el punto de referencia en una
% foto del objeto, debemos tener en cuenta que la foto debe ser
realizada
% con el mismo montaje óptico que los hologramas.

if(tipcamp==1)
```

```
   entradapix1=menu('¿Cómo va introducir el punto de referencia?',...
      'Por coordenadas de pixel','Por localización en imagen del
objeto');
   while(isempty(entradapix1))
      h2=warndlg('Seleccione una opción válida');
      wairfor(h2)
      entradapix1=menu('¿Cómo va introducir el punto de
referencia?',...
         'Por coordenadas de pixel','Por localización en imagen del
objeto');
   end
   entradapix=entradapix1-1;
   if(entradapix==0)          % Elegimos dar las coordenadas del píxel
                              % del punto referencia
      pront={'Componente i:','Componente j:'};
      coorpix1=inputdlg(pront,...
      'Introduzca las componentes i,j del pixel que concuerda con el
punto de referencia:');
      if(isempty(coorpix1))
         return
      end
      while((isempty(coorpix1{1}))|(isempty(coorpix1{2})))
         h3=warndlg('Coordenadas no válidas.');
         waitfor(h3)
         coorpix1=inputdlg(pront,...
     'Introduzca las componentes i,j del pixel que concuerda con el
punto de referencia:');
         if(isempty(coorpix1))
            return
         end
      end
      ay1=eval(coorpix1{1},'-1');
      ay2=eval(coorpix1{2},'-1');
         while((ay1==-1)|(ay2==-1))
            h3=warndlg('Coordenadas no válidas.');
            waitfor(h3)
            coorpix1=inputdlg(pront,...
      'Introduzca las componentes i,j del pixel que concuerda con el
punto de referencia:');
            if(isempty(coorpix1))
               return
            end
            while((isempty(coorpix1{1}))|(isempty(coorpix1{2})))
               h3=warndlg('Coordenadas no válidas.');
               waitfor(h3)
                  coorpix1=inputdlg(pront,...
     'Introduzca las componentes i,j del pixel que concuerda con el
punto de referencia:');
               if(isempty(coorpix1))
                  return
               end
            end
            ay1=eval(coorpix1{1},'-1');
            ay2=eval(coorpix1{2},'-1');
         end
      coorpix=[eval(coorpix1{1}) eval(coorpix1{2})];    % coordenadas
i,j
```

```
                                                          % del píxel
   end

   if(entradapix==1)           % Caso de elegir introducir píxel
                               % localizando en foto
      fotoobj=[];              % Preguntamos al usuario por el nombre
de
                               % la foto.bmp
      cadenafoto=inputdlg(...
         'Introduzca el nombre del archivo bmp que contiene la
fotografía del objeto: ');
      while(isempty(cadenafoto))
         return
      end
      fotoobj=double(imread(cadenafoto{1},'bmp'))/255;
      [i,j]=pincho(fotoobj);   % esta función "pincho" me devuelve la
                               % posición del píxel del
      coorpix=[i j];           % punto de la foto marcado con el
cursor
                               % por el usuario
   end

   % Ahora necesitamos saber el desplazamiento del punto de
referencia;
   % se pide por pantalla.
   prompt={'Introduzca el desplazamiento del punto de referencia(m):
'};
   lineno=1;
   def={'0'};
   despref1=inputdlg(prompt,'Desplazamiento de
referencia',lineNo,def);
   if(isempty(despref1))
      return
   end
   despref=eval(despref1{1});
end




% Preguntamos al usuario por el nombre del archivo bmp donde está
% almacenado el holograma en el estado indeformado (situació inicial).

H1=[];

cadena1=inputdlg('Introduzca el nombre del archivo bmp que representa
el holograma en la situación indeformada: ');
while(isempty(cadena1))
   return
end

H1=double(imread(cadena1{1},'bmp'))/255;       % convierto el tipo de
                                               % datos en class double
```

```
% Ahora preguntamos al usuario por el nombre del archibo bmp donde
está
% almacenado el holograma en el estado deformado

H2=[];

cadena2=inputdlg('Introduzca el nombre del archivo bmp que representa
el holograma en la situación deformada: ');
while(isempty(cadena2))
   return
end

H2=double(imread(cadena2{1},'bmp'))/255;     % convierto el tipo de
datos
                                             % en “class double”.



% Hacemos una comprobación de que los hologramas tienen el mismo
tamaño,
% ya que si no tienen el mismo tamaño me dará error mas adelante
(después
% de esperar el cálculo de la reconstrucción virtual).
% Si no tienen el mismo tamaño no se pueden comparar los dos estados.

[f1,c1]=size(H1);

[f2,c2]=size(H2);

if((f1~=f2)|(c1~=c2))
   errordlg('Los hologramas no son del mismo tamaño.No se pueden
comparar.');
   return
end



% Una vez leídos los hologramas debemos hacer la puntualización
pendiente para el caso de aplicar aproximación FFT.
% Para ajustar la distancia de reconstrucción (y así evitar el
fenómeno
% de aberración) los hologramas deben ser matrices cuadradas.
% Haciendo las operaciones siguientes conseguimos que el tamaño del
píxel
% en plano de reconstrucción sea el mismo que en el de difracción
(sólo
% hacemos esto para FFT).

if(TIPAPROX==0)              % Caso de aproximación FFT
   if(f1>c1)                 % si número de filas mayor que número de
                             % columnas,añado
      H1(:,(c1+1):f1)=0;     % columnas de ceros hasta que sean
                             % cuadradas.
      H2(:,(c1+1):f1)=0;
   end
   if(f1<c1)                 % si número de filas menor que número de
```

```
                              % columnas,añado filas
      H1((f1+1):c1,:)=0;      % de ceros hasta que sean cuadradas.
      H2((f1+1):c1,:)=0;
   end                        % una vez que los hologramas son
cuadrados,
                              % calculo la
   [f,c]=size(H1);            % distancia de reconstrucción para FFT
   d_OBJ_CCD=c*PIXEL^2/LONG_ONDA;
end



% Ahora procedemos a la reconstrucción digital de los hologramas
mediante
% el programa  modificado de R. Coronado Santos.
% Los parámetros de los que depende esta reconstrucción se pasan al
% programa mediante variables globales
% Se reconstruye tanto la situación indeformada como la deformada.
% Para el proceso de reconstrucción hay que tenter en cuenta como se
hizo
% el registro del holograma.
% Si el objeto es cuasi plano y los haces de iluminación son planos
% podemos suponer que se cumplen las condiciones para aplicar FFT. Si
no,
% estaremos obligados a resolver la integral de Fresnel-Kirchoff con
el
% consiguiente gasto de recursos de memoria y tiempo de la
computadora.

HR1=main4(H1);              % reconstrucción de H1, el campo
reconstruido
                            % se almacena en H1R

HR2=main4(H2);              % reconstrucción de H2, el campo
reconstruido
                            % se almacena en H2R



% Una vez hecha la reconstrucción, en el caso de haberse hecho con
FFT,
% podemos quitar las filas o columnas de ceros añadidos para que el
% tamaño de pixel fuera el mismo.

if(TIPAPROX==0)
   if(f1>c1)                  % si número de columnas menor que
número
                              % de filas, quito
      HR1(:,(c1+1):f1)=[];    % columnas de ceros añadidas hasta
                              % recuperar tamaño inicial.
      HR2(:,(c1+1):f1)=[];
   end
   if(f1<c1)                  % si número de columnas mayor que
número
                              % de filas, quito
      HR1((f1+1):c1,:)=[];    % filas de ceros añadidas hasta
recuperar
```

```
                                % tamaño inicial.
      HR2((f1+1):c1,:)=[];
   end
end



% Ahora calculamos la diferencia de fase entre ambos campos
% electromagnéticos reconstruidos

fase1=angle(HR1);                   % fase del campo 1

fase2=angle(HR2);                   % fase del campo2

I=find(fase2(:)<fase1(:));          % buscamos los valores que cumplen
                                    % esto para calcular
                                    % los incrementos de fase
                                    % correctamente.
incremfas=fase2-fase1;

incremfas(I)=incremfas(I)+2*pi;     % aquí ya tenemos calculado el
                                    % "interferograma",es decir
                                    % , los valores de "incremfas"
están
                                    % comprendidos entre
                                    % cero y  2*pi



% Calculamos el incremento de fase relativo,e.d., es la diferencia de
% fase entre punto y punto, pero no es la fase absoluta, para eso
% necesitamos el punto de referencia.
incfasrel=unwrap(incremfas);



% Ahora calculamos el incremento de fase absoluto (en el caso de que
% queramos sólo el relativo no habrá diferencia entre "incfasrel" y
% "incfasabs").
% Modifico la variable aux en el caso de querer campo de
desplazamientos
% absolutos.
% La diferencia de fase absoluta la obtengo comparando la fase
relativa
% del punto de referencia con la fase absoluta calculada con el
% desplazamiento del punto de referencia.

aux=0;

if(tipcamp==1)
   incfasref=despref*vectsensi(3);       % cálculo la fase absoluta
del
                                         % punto de referencia
   aux=incfasrel(coorpix(1),coorpix(2))-incfasref;
end
```

```
incfasabs=incfasrel-aux;            % La diferencia de fase absoluta
para
                                    % todo P se obtiene
                                    % restando la diferencia entre la
fase
                                    % relativa generada
                                    % por unwrap y el incremento de
fase
                                    % absoluto de punto
                                    % referencia (es decir, restando
aux).


% El incremento de fase absoluto queda calculado(en el caso de campo
% relativo aux=0 y el abs=rel)



% Para el calculo del campo de desplazamientos no queda mas que
dividir
% al diferencia de fase absoluta por la componente zeta del vector de
% sensibilidad.
% El campo de desplazamientos  se almacena en Dz que es el argumento
de
% retorno que devuelve a la función "calcdesp" para el postprocesado.

Dz=incfasabs/vectsensi(3);          % El campo de desplazamientos queda
así
                                    % calculado

opcfinal(Dz);          % Llamamos a esta función para procesar el campo
                       % calculado.
```

```
%                              calc3D.m


%        Programa realizado por Javier González Cuenca


function [Dx,Dy,Dz]=calc3D

% Esta función calculará el campo de desplazamientos tridimensional.
% Para ello se deben haber hecho en el laboratorio tres
% "interferogramas", es decir, se comparará la situación indeformada y
% deformada según tres direcciones de iluminación distintas.
% En total se deben haber hecho registros de seis hologramas, dos por
% cada dirección de iluminación.
% Estas direcciones de iluminación deben formar base, para poder
obtener
% información 3D.



global d_OBJ_CCD LONG_ONDA PIXEL E0_ONDA TIPAPROX


Dz=[];               % Aqui almacenaremos los desplazamientos en la
                     % dirección z de cada pixel.

Dx=[];               % dirección x

Dy=[];               % dirección y



% Se pregunta al usuario por el tipo de montaje.

tipsens1=menu('¿Su montaje es con vector de sensibilidad
constante?','Sí','No');

while(isempty(tipsens1))
   h2=warndlg('Seleccione una opción válida');
   wairfor(h2)
   tipsens1=menu('¿Su montaje es con vector de sensibilidad
constante?','Sí','No');
end

tipsens=tipsens1-1;

if(tipsens==1)
   errordlg('No disponible para vector de sensibilidad variable')
   return
end
```

```
% A continuación se pedirán por pantalla las componentes de los
vectores
% de iluminación.

% Dirección I

pront={'Componente x:','Componente y:','Componente z:'};

vectorilu1=inputdlg(pront,'Vector dirección de iluminación I :');

if(isempty(vectorilu1))
   return
end

while((isempty(vectorilu1{1}))|(isempty(vectorilu1{2}))|(isempty(vecto
rilu1{3})))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu1=inputdlg(pront,'Vector dirección de iluminación I :');
   if(isempty(vectorilu1))
      return
   end
end

ay1=eval(vectorilu1{1},'-1*j');
ay2=eval(vectorilu1{2},'-1*j');
ay3=eval(vectorilu1{3},'-1*j');

while((ay1==-1*j)|(ay2==-1*j)|(ay3==-1*j))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu1=inputdlg(pront,'Vector dirección de iluminación I :');
   if(isempty(vectorilu1))
      return
   end

while((isempty(vectorilu1{1}))|(isempty(vectorilu1{2}))|(isempty(vecto
rilu1{3})))
      h3=warndlg('Vector no válido.');
      waitfor(h3)
      vectorilu1=inputdlg(pront,'Vector dirección de iluminación');
      if(isempty(vectorilu1))
         return
      end
   end
   ay1=eval(vectorilu1{1},'-1*j');
   ay2=eval(vectorilu1{2},'-1*j');
   ay3=eval(vectorilu1{3},'-1*j');
end

vectoriluI=[eval(vectorilu1{1}) eval(vectorilu1{2})
eval(vectorilu1{3})];

moduloiluI=sqrt(vectoriluI(1)^2+vectoriluI(2)^2+vectoriluI(3)^2);
   % normalizo la dirección de iluminación I,por si acaso se ha
   % introducido sin normalizar.
```

```
vectoriluI=vectoriluI/moduloiluI;
   % el vector de iluminación I queda así calculado.



% Dirección II

pront={'Componente x:','Componente y:','Componente z:'};

vectorilu2=inputdlg(pront,'Vector dirección de iluminación II :');

if(isempty(vectorilu2))
   return
end

while((isempty(vectorilu2{1}))|(isempty(vectorilu2{2}))|(isempty(vecto
rilu2{3})))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu1=inputdlg(pront,'Vector dirección de iluminación II :');
   if(isempty(vectorilu2))
      return
   end
end

ay1=eval(vectorilu2{1},'-1*j');
ay2=eval(vectorilu2{2},'-1*j');
ay3=eval(vectorilu2{3},'-1*j');

while((ay1==-1*j)|(ay2==-1*j)|(ay3==-1*j))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu1=inputdlg(pront,'Vector dirección de iluminación II :');
   if(isempty(vectorilu2))
      return
   end

while((isempty(vectorilu2{1}))|(isempty(vectorilu2{2}))|(isempty(vecto
rilu2{3})))
      h3=warndlg('Vector no válido.');
      waitfor(h3)
      vectorilu2=inputdlg(pront,'Vector dirección de iluminación II
:');
      if(isempty(vectorilu2))
         return
      end
   end
   ay1=eval(vectorilu2{1},'-1*j');
   ay2=eval(vectorilu2{2},'-1*j');
   ay3=eval(vectorilu2{3},'-1*j');

end

vectoriluII=[eval(vectorilu2{1}) eval(vectorilu2{2})
eval(vectorilu2{3})];
```

```
moduloiluII=sqrt(vectoriluII(1)^2+vectoriluII(2)^2+vectoriluII(3)^2);
   % normalizo la dirección de iluminación II,por si acaso se ha
   % introducido sin normalizar.

vectoriluII=vectoriluII/moduloiluII;
   % el vector de iluminación II queda así calculado.



% Dirección III

pront={'Componente x:','Componente y:','Componente z:'};

vectorilu3=inputdlg(pront,'Vector dirección de iluminación III :');

if(isempty(vectorilu3))
   return
end

while((isempty(vectorilu3{1}))|(isempty(vectorilu3{2}))|(isempty(vecto
rilu3{3})))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu3=inputdlg(pront,'Vector dirección de iluminación III :');
   if(isempty(vectorilu3))
      return
   end
end

ay1=eval(vectorilu3{1},'-1*j');
ay2=eval(vectorilu3{2},'-1*j');
ay3=eval(vectorilu3{3},'-1*j');

while((ay1==-1*j)|(ay2==-1*j)|(ay3==-1*j))
   h3=warndlg('Vector no válido.');
   waitfor(h3)
   vectorilu3=inputdlg(pront,'Vector dirección de iluminación III :');
   if(isempty(vectorilu3))
      return
   end

while((isempty(vectorilu3{1}))|(isempty(vectorilu3{2}))|(isempty(vecto
rilu3{3})))
      h3=warndlg('Vector no válido.');
      waitfor(h3)
      vectorilu3=inputdlg(pront,'Vector dirección de iluminación III
:');
      if(isempty(vectorilu3))
         return
      end
   end
   ay1=eval(vectorilu3{1},'-1*j');
   ay2=eval(vectorilu3{2},'-1*j');
   ay3=eval(vectorilu3{3},'-1*j');

end
```

```
vectoriluIII=[eval(vectorilu3{1}) eval(vectorilu3{2})
eval(vectorilu3{3})];

moduloiluIII=sqrt(vectoriluIII(1)^2+vectoriluIII(2)^2+vectoriluIII(3)^
2);
   % normalizo la dirección de iluminación III,por si acaso se ha
   % introducido sin normalizar.

vectoriluIII=vectoriluIII/moduloiluIII;
   % el vector de iluminación III queda así calculado.



% Ahora calculamos los vectores de sensibilidad

vectorobs=[0 0 1];         % el vector de observación siempre es el
                           % mismo, normal al CCD

vectsensiI=(2*pi/LONG_ONDA)*(vectorobs-vectoriluI);
                           % calculo del vector de sensibilidad según
I

vectsensiII=(2*pi/LONG_ONDA)*(vectorobs-vectoriluII);
                           % calculo del vector de sensibilidad según
II

vectsensiIII=(2*pi/LONG_ONDA)*(vectorobs-vectoriluIII);
                           % calculo del vector de sensibilidad según
III







% Preguntamos si el campo de desplazamientos a calcular es absoluto o
% relativo

tipcamp1=menu('El campo de desplazamientos a calcular
es:','Relativo','Absoluto');

while(isempty(tipcamp1))
   h2=warndlg('Seleccione una opción válida');
   wairfor(h2)
   tipcamp1=menu('El campo de desplazamientos a calcular
es:','Relativo','Absoluto');
end

tipcamp=tipcamp1-1;        % si el campo que queremos calcular es
                           % absoluto tenemos que
                           % introducir al desplazamiento y
                           % coordenadas(en píxel, es decir,
                           % en el plano del CCD) del punto de
referencia
```

```
% Para el caso de campo de desplazamientos absoluto necesitamos
% introducir  el punto de referencia y su desplazamiento.
% Este punto de referencia del objeto concuerda con un píxel del CCD
% (según la dirección de observación).Podemos reconocer este punto
% mediante cambio de sistemas de referencia (de objeto a CCD) o
% localizándolo en una foto del objeto (foto, no holograma)hecha con
el
% CCD en la posición del experimento.

if(tipcamp==1)   entradapix1=menu('¿Como va introducir el punto de
referencia?',...
      'Por coordenadas de pixel','Por localización en imagen del
objeto');
   while(isempty(entradapix1))
      h2=warndlg('Seleccione una opción válida');
      wairfor(h2)
      entradapix1=menu('¿Como va introducir el punto de
referencia?',...
         'Por coordenadas de pixel','Por localización en imagen del
objeto');
   end
   entradapix=entradapix1-1;
   if(entradapix==0)                  % pedimos la coordenadas del
píxel
                                      % por pantalla.
      pront={'Componente i:','Componente j:'};
      coorpix1=inputdlg(pront,...
         'Introduzca las componentes i,j del pixel que concuerda con
el punto de referencia:');
      if(isempty(coorpix1))
         return
      end
      while((isempty(coorpix1{1}))|(isempty(coorpix1{2})))
         h3=warndlg('Coordenadas no válidas.');
         waitfor(h3)
         coorpix1=inputdlg(pront,...
            'Introduzca las componentes i,j del pixel que concuerda
con el punto de referencia:');
         if(isempty(coorpix1))
            return
         end
      end
      ay1=eval(coorpix1{1},'-1');
      ay2=eval(coorpix1{2},'-1');
         while((ay1==-1)|(ay2==-1))
            h3=warndlg('Coordenadas no válidas.');
            waitfor(h3)
            coorpix1=inputdlg(pront,...
               'Introduzca las componentes i,j del pixel que concuerda
con el punto de referencia:');
            if(isempty(coorpix1))
               return
            end
```

```
            while((isempty(coorpix1{1}))|(isempty(coorpix1{2})))
               h3=warndlg('Coordenadas no válidas.');
               waitfor(h3)
                  coorpix1=inputdlg(pront,...
                     'Introduzca las componentes i,j del pixel que
concuerda con el punto de referencia:');
               if(isempty(coorpix1))
                  return
               end
            end
            ay1=eval(coorpix1{1},'-1');
            ay2=eval(coorpix1{2},'-1');
         end
         coorpix=[eval(coorpix1{1}) eval(coorpix1{2})];
      end                    % aquí tengo las coordenadas i,j del píxel
                             % que corresponde
                             % (según la dirección de observación) con
el
                             % punto de referencia

                             % Si vamos a introducirlo mediante
                             % localización en la foto del
   if(entradapix==1)         % objeto, necesitamos saber donde esta
                             % almacenada dicha foto,
      fotoobj=[];            % hecha en las mismas condiciones que los
                             % hologramas.
      cadenafoto=inputdlg('Introduzca el nombre del archivo bmp que
representa la fotografía del objeto: ');
      while(isempty(cadenafoto))
         return
      end
      fotoobj=double(imread(cadenafoto{1},'bmp'))/255;
      [i,j]=pincho(fotoobj);    % esta función  "pincho" me devuelve
la
                                % posición del píxel
      coorpix=[i j];            % del punto de la foto marcado con el
                                % cursor por el usuario
    end


   % Ahora necesitamos saber el desplazamiento del punto de
referencia.
   % Se pregunta por pantalla.

   lineNo=1;

   def={'0','0','0'};

   pront={'Componente x:','Componente y:','Componente z:'};

   despref1=inputdlg(pront,'Desplazamineto del punto de
referencia:',lineNo,def);

   if(isempty(despref1))
      return
   end
```

```
while((isempty(despref1{1}))|(isempty(despref1{2}))|(isempty(despref1{
3})))
      h3=warndlg('Vector no válido.');
      waitfor(h3)
      despref1=inputdlg(pront,'Despazamiento del punto de
referencia:',lineNo,def);
      if(isempty(despref1))
         return
      end
   end

   ay1=eval(despref1{1},'-1*j');
   ay2=eval(despref1{2},'-1*j');
   ay3=eval(despref1{3},'-1*j');

   while((ay1==-1*j)|(ay2==-1*j)|(ay3==-1*j))
      h3=warndlg('Vector no válido.');
      waitfor(h3)
      despref1=inputdlg(pront,'Desplazamiento del punto de
referencia:',lineNo,def);
      if(isempty(despref1))
         return
      end

   while((isempty(despref1{1}))|(isempty(despref1{2}))|(isempty(despre
f1{3})))
         h3=warndlg('Vector no válido.');
         waitfor(h3)
         despref1=inputdlg(pront,'Desplazamiento del punto de
referencia:',lineNo,def);
         if(isempty(despref1))
            return
         end
      end
      ay1=eval(despref1{1},'-1*j');
      ay2=eval(despref1{2},'-1*j');
      ay3=eval(despref1{3},'-1*j');

   end
   despref=[eval(despref1{1}) eval(despref1{2}) eval(despref1{3})];
end




% Ahora pregunto por los nombres de los archivos bmp en donde están
% registrados los hologramas

h1=helpdlg(...
   'A continuación se pedirán por pantalla los nombres de los archivos
bmp en donde están registrados los hologramas.Estos hologramas deben
tener el mismo tamaño, al haber sido grabados con el mismo CCD.');

waitfor(h1);
```

```
% Situación indeformada

pront={'Dirección I:','Dirección II:','Dirección III:'};

nombholg1=inputdlg(pront,'Hologramas situación indeformada:');

if(isempty(nombholg1))
   return
end

while((isempty(nombholg1{1}))|(isempty(nombholg1{2}))|(isempty(nombhol
g1{3})))
   h3=warndlg('Nombres no válidos.');
   waitfor(h3)
   nombholg1=inputdlg(pront,'Hologramas situación indeformada:');
   if(isempty(nombholg1))
      return
   end
end

H1I=double(imread(nombholg1{1},'bmp'))/255;     % nombre del holograma
en
                                                % situación indeformada
                                                % según la dirección I

H1II=double(imread(nombholg1{2},'bmp'))/255;    % nombre del holograma
en
                                                % situación indeformada
                                                % según la dirección I

H1III=double(imread(nombholg1{3},'bmp'))/255;   % nombre del holograma
en
                                                % situación indeformada
                                                % según la dirección I



% Situación deformada.

pront={'Dirección I:','Dirección II:','Dirección III:'};

nombholg2=inputdlg(pront,'Hologramas situación deformada:');

if(isempty(nombholg2))
   return
end

while((isempty(nombholg2{1}))|(isempty(nombholg2{2}))|(isempty(nombhol
g2{3})))
   h3=warndlg('Nombres no válidos.');
   waitfor(h3)
   nombholg2=inputdlg(pront,'Hologramas situación deformada:');
   if(isempty(nombholg2))
      return
   end
end
```

```
H2I=double(imread(nombholg2{1},'bmp'))/255;     % nombre del holograma
en
                                                % situación deformada
                                                % según la dirección I

H2II=double(imread(nombholg2{2},'bmp'))/255;    % nombre del holograma
en
                                                % situación deformada
                                                % según la dirección II

H2III=double(imread(nombholg2{3},'bmp'))/255;   % nombre del holograma
en
                                                % situación deformada
                                                % según la dirección
III



% Comprobación de que las matrices tienen el mismo tamaño.
% Si no tienen el mismo tamaño no se pueden camparar. En caso de tener
el
% mismo tamaño mostraremos un mensaje de error y el programa se
parará.

[f1I,c1I]=size(H1I);                % tamaño de los hologramas; los
                                    % índices de la fila y la
                                    % columna de cada holograma
coinciden
                                    % con los de su holograma.
[f1II,c1II]=size(H1II);

[f1III,c1III]=size(H1III);

[f2I,c2I]=size(H2I);

[f2II,c2II]=size(H2II);

[f2III,c2III]=size(H2III);

if((f1I~=f1II)|(f1I~=f1III)|(f2I~=f2II)|(f2I~=f2III)|(f1I~=f2I)|(c1I~=
c1II)|(c1I~=c1III)|(c1I~=c2I)|(c2I~=c2II)|(c2I~=c2III))
   errordlg('Los hologramas no son del mismo tamaño. No se pueden
comparar.');
   return
end



% Una vez leidos los hologramas debemos hacer la puntualización
pendiente
% para el caso de poder aplicar FFT.
% Para ajustar la distancia de reconstrucción (y así evitar el
fenómeno
% de aberración) los hologramas deben ser matrices cuadradas.

if(TIPAPROX==0)
```

```
   [f1,c1]=size(H1I);            % si los hologramas ya son cuadrados
no
                                 % se modifican
   if(f1>c1)                     % si número de columnas menor que
número
                                 % de filas,
      H1I(:,(c1+1):f1)=0;        % añado columnas de ceros en todos
los
                                 % hologramas
      H1II(:,(c1+1):f1)=0;       % hasta que sean cuadrados.
      H1III(:,(c1+1):f1)=0;
      H2I(:,(c1+1):f1)=0;
      H2II(:,(c1+1):f1)=0;
      H2III(:,(c1+1):f1)=0;
   end
   if(f1<c1)                     % si número de filas menor que número
de
                                 % filas,
      H1I((f1+1):c1,:)=0;        % añado filas de ceros en todos los
                                 % hologramas hasta que
      H1II((f1+1):c1,:)=0;       % sean cuadrados.
      H1III((f1+1):c1,:)=0;
      H2I((f1+1):c1,:)=0;
      H2II((f1+1):c1,:)=0;
      H2III((f1+1):c1,:)=0;
   end                               % Una vez que los hologramas son
                                     % cuadrados, calculo la
   [f,c]=size(H1I);                  % distancia de reconstrucción
para
                                     % FFT.
   d_OBJ_CCD=c*PIXEL^2/LONG_ONDA;    % Cálculo de la distancia a la
que
                                     % evito la aberración;
end                                  % consigo mismo tamaño de pixel
en
                                     % holograma reconstruido




% Procedemos a la reconstrucción de los hologramas para comparar los
% campos electromagnéticos reconstruidos.

HR1I=main4(H1I);

HR1II=main4(H1II);

HR1III=main4(H1III);

HR2I=main4(H2I);

HR2II=main4(H2II);

HR2III=main4(H2III);
```

```
% Una vez reconstruidos los campos electromagnéticos, podemos eliminar
% las filas o columnas de ceros añadidos en el caso de reconstrucción
% FFT.

if(TIPAPROX==0)
   if(f1>c1)                        % si número de filas mayor que
número
                                    % de columnas,
      HR1I(:,(c1+1):f1)=[];         % elimino columnas añadidas.
      HR1II(:,(c1+1):f1)=[];
      HR1III(:,(c1+1):f1)=[];
      HR2I(:,(c1+1):f1)=[];
      HR2II(:,(c1+1):f1)=[];
      HR2III(:,(c1+1):f1)=[];
   end
   if(f1<c1)                        % si número de columnas mayor que
                                    % número de filas,
      HR1I((f1+1):c1,:)=[];         % elimino filas añadidas.
      HR1II((f1+1):c1,:)=[];
      HR1III((f1+1):c1,:)=[];
      HR2I((f1+1):c1,:)=[];
      HR2II((f1+1):c1,:)=[];
      HR2III((f1+1):c1,:)=[];
   end
end



% Ahora calculamos la diferencia de fase entre ambos campos
% electromagnéticos reconstruidos


% Dirección I

fase1I=angle(HR1I);    % fase campo reconstruido según la dirección I
en
                       % el estado indeformado

fase2I=angle(HR2I);    % fase campo reconstruido según la dirección I
en
                       % el estado deformado

I=find(fase2I(:)<fase1I(:));

incremfasI=fase2I-fase1I;

incremfasI(I)=incremfasI(I)+2*pi;           % aquí ya tenemos
calculado
                                            % el "interferograma I"

incfasrelI=unwrap(incremfasI);
                     % calculamos el incremento de fase relativo I
                     % ,e.d., es la diferencia de fase entre punto y
```

```
                    % punto, pero no es la fase absoluta para eso
                    % necesitamos el punto de referencia.


% Dirección II

fase1II=angle(HR1II);  % fase campo reconstruido según la dirección II
en
                       % el estado indeformado

fase2II=angle(HR2II);  % fase campo reconstruido según la dirección II
en
                       % el estado deformado

I=find(fase2II(:)<fase1II(:));

incremfasII=fase2II-fase1II;

incremfasII(I)=incremfasII(I)+2*pi;    % aquí ya tenemos calculado el
                                       % "interferograma" II

incfasrelII=unwrap(incremfasII);
                       % calculamos el incremento de fase relativo II
                       % ,e.d., es la
                       % diferencia de fase entre punto y punto, pero
no
                       % es la fase absoluta
                       % para eso necesitamos el punto de referencia.




% Dirección III

fase1III=angle(HR1III);           % fase campo reconstruido según la
                                  % dirección III en
                                  % el estado indeformado.

fase2III=angle(HR2III);           % fase campo reconstruido según la
                                  % dirección III en
                                  % el estado deformado.

I=find(fase2III(:)<fase1III(:));

incremfasIII=fase2III-fase1III;

incremfasIII(I)=incremfasIII(I)+2*pi;    % aquí ya tenemos calculado
el
                                         % "interferograma" III

incfasrelIII=unwrap(incremfasIII);
                       % calculamos el incremento de fase relativo III
                       % ,e.d., es la diferencia de fase entre punto y
                       % punto, pero no es la fase absoluta para eso
                       % necesitamos el punto de referencia.
```

```
% Ahora calculamos el incremento de fase absoluto (en el caso de que
% queramos sólo el relativo no habrá diferencia entre "incfasrel" y
% "incfasabs".

auxI=0;

auxII=0;

auxIII=0;

G=[vectsensiI;vectsensiII;vectsensiIII];     % Aquí hemos calculado la
                                             % matriz de sensibilidad.


if(tipcamp==1)               % Si el tipo de campo es absoluto,
calculo
                             % la diferencia de fase
   incfasref=G*despref';     % entre la fase del punto de referencia y
la
                             % fase relativa.
   auxI=incfasrelI(coorpix(1),coorpix(2))-incfasref(1);
   auxII=incfasrelII(coorpix(1),coorpix(2))-incfasref(2);
   auxIII=incfasrelIII(coorpix(1),coorpix(2))-incfasref(3);
end

incfasabsI=incfasrelI-auxI;              % el incremento de fase
                                         % absoluto I queda calculado


incfasabsII=incfasrelII-auxII;           % el incremento de fase
                                         % absoluto II queda
calculado

incfasabsIII=incfasrelIII-auxIII;        % el incremento de fase
                                         % absoluto III queda
calculado



% Procedemos a calcular el campo de desplazamientos 3D.
% El campo de desplazamientos calculado se devuelve a la función de
% llamada "calcdesp" para el procesado del campo.
% El campo de desplazamientos se calcula cada componente por separado.
% De esta forma el campo queda almacenado tres matrices, cada una con
las
% componentes x,y,z respectivamente del campo vectorial de
% desplazamientos.

F=inv(G);                        % invertimos la matriz de
sensibilidad


% Componente x de los vectores del campo de desplazaminetos.
```

```
Dx=F(1,1)*incfasabsI+F(1,2)*incfasabsII+F(1,3)*incfasabsIII;

% Componente y de los vectores del campo de desplazaminetos.

Dy=F(2,1)*incfasabsI+F(2,2)*incfasabsII+F(2,3)*incfasabsIII;

% Componente z de los vectores del campo de desplazaminetos.

Dz=F(3,1)*incfasabsI+F(3,2)*incfasabsII+F(3,3)*incfasabsIII;


opcfinal(Dx,Dy,Dz);    % Llamamos a la función para procesar el campo
                       % calculado.
```

```
%                           pincho.m

%            Programa realizado por Javier González Cuenca


function [filapix,columpix]=pincho(L)


% Esta función me permite seleccionar un punto de la fotografía del
% objeto que le paso a la función como la matriz L. Me devuelve dos
% valores enteros correspondientes a las coordenadas i,j del píxel
% marcado en último lugar por el usuario. El origen de estos valores
está
% en la esquina superior izquierda, de tal forma que si el usuario
% marcara esta posición, la función devolvería los valores [1 1]

close all;

h1=helpdlg('A continuación se le mostrará una imagen del objeto. Pulse
sobre ella la posición del punto de referencia. Cuando este seguro de
haberlo localizado, pulse "intro"');

waitfor(h1);

% Mostramos la foto al usuario
figure,imshow(L)

[x,y]=ginput;     % La función ginput me devuelve dos vectores con las
                  % posiciones x,y de todos los intentos realizados
por
                  % el usuario 0on el cursor sobre la foto.
                  % Los valores y,x tiene el mismo origen y dirección
                  % que los i,j de la foto.

[n,m]=size(x);    % Me quedo con el resultado del último intento, es
                  % decir, de todos los valores almacenados en x,y
solo
                  % me interesa el último; lo calculo indexando
                  % directamente sobre el último elemento habiendo
                  % calculado previamente el tamaño de x,y con la
función
                  % [n,m]=size(x);

columpix=x(n);    % En una imagen, la coordenada x es la columna, y la
                  % coordenada y es la fila, y así los valores de
retorno
                  % del programa quedan calculados.

filapix=y(n);     % columpix es la columna del pixel del punto de
                  % referencia
                  % filapix es la fila del pixel del punto de
referencia.
```

```
% Segun el sistema de coordenadas en una imagen, tenemos que hacer
% redondear al número entero más cercano, ya que los píxel i,j se
% corresponden con las coordenadas y,x de valores comprendidos
% entre [ (y1=i-0.5) < i < (y2=i+0.5) ] y [ (x=j-0.5) < j < (x=j+0.5)
]
% Asi los valores no enteros devueltos por la función ginput son
% transformados a unidades píxel, mas propias para trabajar con
imágenes
% y matrices.

columpix=round(columpix);

filapix=round(filapix);



% Antes de devolver los valores calculados, comprobamos que están
dentro
% del tamaño de la foto del objeto. Si no cumple este mostramos un
% mensaje de error y pedimos al usuario que vuelva a repetir la
% operación.

[f,c]=size(L);

close;

while((filapix<1)|(filapix>f)|(columpix<1)|(columpix>c))
   h2=errordlg('Error, el punto seleccionado tiene que estar dentro de
la foto. Intentelo otra vez y luego, pulse enter.');
   waitfor(h2);
   figure,imshow(L)
   [x,y]=ginput;
   [n,m]=size(x);
   columpix=x(n);
   filapix=y(n);
   columpix=round(columpix);
   filapix=round(filapix);
   close;
end
```

```
%                       opcfinal.m


%               Programa realizado por Javier González Cuenca



function opcfinal(varargin)

% Función que permite seleccionar entre distintos tipos de
posibilidades
% lo que queremos hacer con los resultados obtenidos en otros
programas.
% Puede tener distintos argumentos de retorno. Si tiene un argumento
de
% retorno al elegir una de las opciones del menú inicial, llama a
% funciones para el postprocesado de un campo monodimensional. Si
tiene
% tres argumentos llama a funciones de posrprocesado de campos 3D.



h1=helpdlg('El campo de desplazamientos ha sido calculado. A
continuación se le mostrarán las distintas opciones de que dispone.');

waitfor(h1);

opciones=menu('Elija una opción. ¿Qué desea hacer?',...
   'Visualizar resultados',...
   'Guardar el campo de desplazamientos',...
   'Postprocesado del campo',...
   'Terminar');

while(isempty(opciones))
   h2=warndlg('Seleccione una opción válida');
   waitfor(h2)
   opciones=menu('Elija una opción. ¿Qué desea hacer?',...
      'Visualizar resultados',...
      'Guardar el campo de desplazamientos',...
      'Terminar');
end


% Una vez seleccionada una de las opciones del menú se ejecutan las
% siguientes sentencias de la opción elegida.

% Visualizar resultados.
if(opciones==1)
   if(nargin==1)
      vermono(varargin{1});
   end
   if(nargin==3)
      ver3D(varargin{1},varargin{2},varargin{3})
   end
end
```

```
% Guardar el campo de desplazamientos.
if(opciones==2)
    if(nargin==1)
      save varargin{1};
   end
   if(nargin==3)
      save varargin{1}, varargin{2}, varargin{3} ;
   end
end


% Postprocesado del campo. El postprecesado del campo podría ser el
% calculo de tensiones y deformaciones. Sin embargo, no es posible
% calcular esto sin conocer las propiedades del material, condiciones
de
% carga, geometría del problema, etc...
% Solo en unos pocos casos concretos como el de una laja, tensión
plana o
% vigas, de materiales conocidos y con condiciones de contorno muy
% limitadas, se podría hacer un postprocesado inmediato de las
tensiones
% y deformaciones del sólido, al ser el campo de tensiones deducido a
% partir de los desplazamientos de la superficie del objeto único.
% De momento esta posibilidad no está disponible. Ofrecemos la
% posibilidad de guardar las variables y exportarlas a programas más
% especificos.


if(opciones==3)
   h3=errordlg('Postprocesado no disponible');
   waitfor(h3);
end

% Terminar

if(opciones==4)
   return
end

% Hasta que no se pulse terminar, el menú de opciones permanecerá en
% pantalla, a la espera del usuario.

aux=0;

while(aux==0)
   opciones=menu('Elija una opción. ¿Qué desea hacer?',...
      'Visualizar resultados',...
      'Guardar el campo de desplazamientos',...
      'Postprocesado del campo',...
      'Terminar');
   while(isempty(opciones))
      h2=warndlg('Seleccione una opción válida');
      waitfor(h2);
      opciones=menu('Elija una opción. ¿Qué desea hacer?',...
         'Visualizar resultados',...
         'Guardar el campo de desplazamientos',...
         'Postprocesado del campo',...
```

```
          'Terminar');
   end

   if(opciones==1)
      if(nargin==1)
         vermono(varargin{1});
      end
      if(nargin==3)
         ver3D(varargin{1},varargin{2},varargin{3})
      end
   end
   if(opciones==2)
      if(nargin==1)
          Dz=varargin{1};
          save Dz;
      end
      if(nargin==3)
         Dx=varargin{1};
         Dy=varargin{2};
         Dz=varargin{3};
         save Dx, Dy, Dz;
      end
   end
   if(opciones==3)
      h4=errordlg('Postprocesado no disponible');
      waitfor(h4);
   end
   if(opciones==4)
      return
   end
end
```

```
%                    vermono.m


%       Programa realizado por Javier González Cuenca


function vermono(Dz)

% Función para visualizar resultados del calculo monodimensional del
% campo de desplazamientos.
% Recibe como argumento de entrada la matriz Dz, en donde están
almacena
% dos los desplazamientos en la dirección z.
% Puede ser utilizada en otro tipo de programas para la visualización
% gráfica de una matriz cualquiera.


auxmono=0;

while(auxmono==0)
   grafico=menu('¿Cómo va a visualizar los resultados?',...
      'En ejes tridimensionales',...
      'Como mapa de colores',...
      'Curvas de nivel',...
      'Volver al menú principal');
   while(isempty(grafico))
      h2=warndlg('Seleccione una opción válida');
      wairfor(h2)
      grafico=menu('¿Cómo va a visualizar los resultados?',...
         'En ejes tridimensionales',...
         'Como mapa de colores',...
         'Curvas de nivel',...
         'Volver al menú principal');
   end

   close all;



   % Ejes tridimensionales.

   if(grafico==1)
      close all;
      colormap(jet)
      figure(gcf),surf(Dz),colorbar
      axis off
      axis ij
      axis equal
      axis tight

   end


   % Mapa de colores.

   if(grafico==2)
```

```
      close all;
      colormap(jet)
      figure(gcf),imagesc(Dz),colorbar
      axis off
      axis ij
      axis equal
      axis tight

   end



   % Curvas de nivel.

   if(grafico==3)
      close all;
      colormap(jet)
      figure(gcf),contour(Dz),colorbar
      axis off
      axis ij
      axis equal
      axis tight
   end


   % Volver al menú pricipal.

   if(grafico==4)
      return
   end
end
```

```
%                          ver3D.m

%      Programa realizado por Javier González Cuenca.


function ver3D(Dx,Dy,Dz)


% Función que me permite visualizar los campos de desplazamientos
% anteriormente calculados
% Podemos elegir entre varias opciones a la hora de representar los
% campos.
% Recibe como argumento de entrada tres matrices Dx,Dy,Dz que
representan
% las componentes x,y,z del campo de desplazamientos.


aux3D=0;


while(aux3D==0)
   grafico=menu('¿Cómo va a visualizar los resultados?',...
      'En ejes tridimensionales los tres campos',...
      'Como mapa de colores de los tres campos',...
      'Curvas de nivel de los tres campos',...
      'Ver sólo desplazamientos en dirección x',...
      'Ver sólo desplazamientos en dirección y',...
      'Ver sólo desplazamientos en dirección z',...
      'Volver al menú principal');
   while(isempty(grafico))
      h2=warndlg('Seleccione una opción válida');
      wairfor(h2)
      grafico=menu('¿Cómo va a visualizar los resultados?',...
         'En ejes tridimensionales los tres campos',...
         'Como mapa de colores de los tres campos',...
         'Curvas de nivel de los tres campos',...
         'Ver sólo desplazamientos en dirección x',...
         'Ver sólo desplazamientos en dirección y',...
         'Ver sólo desplazamientos en dirección z',...
         'Volver al menú principal');
   end


   close all;







   % En ejes tridimensionales los tres campos

   if(grafico==1)
```

```
   colormap(jet)
   subplot(2,2,1)
   figure(gcf),surf(Dx),colorbar
   title('Dx = Desplazamiento eje x')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,2)
   figure(gcf),surf(Dy),colorbar
   title('Dy = Desplazamiento eje y')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,3)
   figure(gcf),surf(Dz),colorbar
   title('Dz = Desplazamiento eje z')
   axis off
   axis ij
   axis equal
   axis tight

end



% Mapa de colores los tres campos

if(grafico==2)
   colormap(jet)
   subplot(2,2,1)
   figure(gcf),imagesc(Dx),colorbar
   title('Dx = Desplazamiento eje x')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,2)
   figure(gcf),imagesc(Dy),colorbar
   title('Dy = Desplazamiento eje y')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,3)
   figure(gcf),imagesc(Dz),colorbar
   title('Dz = Desplazamiento eje z')
   axis off
   axis ij
   axis equal
   axis tight
```

```
end


% Curvas de  nivel los tres campos.

if(grafico==3)
   colormap(jet)

   subplot(2,2,1)
   figure(gcf),contour(Dx),colorbar
   title('Dx = Desplazamiento eje x')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,2)
   figure(gcf),contour(Dy),colorbar
   title('Dy = Desplazamiento eje y')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,3)
   figure(gcf),contour(Dz),colorbar
   title('Dz = Desplazamiento eje z')
   axis off
   axis ij
   axis equal
   axis tight
end


% Sólo el campo Dx.

if(grafico==4)
   colormap(jet)

   subplot(2,2,1)
   figure(gcf),surf(Dx),colorbar
   title('Dx en ejes tridimensionales')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,2)
   figure(gcf),imagesc(Dx),colorbar
   title('Dx como mapa de colores')
   axis off
   axis ij
   axis equal
   axis tight
```

```
   subplot(2,2,3)
   figure(gcf),contour(Dx),colorbar
   title('Dx como curvas de nivel')
   axis off
   axis ij
   axis equal
   axis tight
end



% Sólo el campo Dy.

if(grafico==5)
   colormap(jet)

   subplot(2,2,1)
   figure(gcf),surf(Dy),colorbar
   title('Dy en ejes tridimensionales')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,2)
   figure(gcf),imagesc(Dy),colorbar
   title('Dy como mapa de colores')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,3)
   figure(gcf),contour(Dy),colorbar
   title('Dy como curvas de nivel')
   axis off
   axis ij
   axis equal
   axis tight
end


% Sólo el campo Dz.

if(grafico==6)
   colormap(jet)

   subplot(2,2,1)
   figure(gcf),surf(Dz),colorbar
   title('Dz en ejes tridimensionales')
   axis off
   axis ij
   axis equal
   axis tight

   subplot(2,2,2)
   figure(gcf),imagesc(Dz),colorbar
```

```
      title('Dz como mapa de colores')
      axis off
      axis ij
      axis equal
      axis tight

      subplot(2,2,3)
      figure(gcf),contour(Dz),colorbar
      title('Dz como curvas de nivel')
      axis off
      axis ij
      axis equal
      axis tight
   end



   % Volver al menú pricipal.

   if(grafico==7)
      return
   end

end
```