Anexos

Modelo.m

```
function varargout = Modelo(varargin)
%clc
%      Modelo M-file for modelo.fig
%      Modelo, by itself, creates a new MODELOor raises the existing
%      singleton*.
%
%      H = Modelo returns the handle to a new MODELOor the handle to
%      the existing singleton*.
%
%      Modelo ('Property','Value',...) creates a new MODELOusing the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to modelo_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      Modelo ('CALLBACK') and Modelo ('CALLBACK',hObject,...) call the
%      local function named CALLBACK in Modelo.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help modelo

% Last Modified by GUIDE v2.5 15-Dec-2011 17:41:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @modelo_OpeningFcn, ...
                   'gui_OutputFcn',  @modelo_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
  gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before modelois made visible.
function modelo_OpeningFcn(hObject, ~, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% PONER IMAGEN DE FONDO
%handles.background=rbg2gray(imread('Boya-2-cerca-de-taladro.jpg'));    % Leer la
imagen
%axes(handles.background);                                    % Carga la imagen en
background
%axes('units','normalized','position', [0 0 1 1]);
%axis off;
%imshow(handles.background);                                   % Muestra la imagen
%[x,map]=rbg2gray(imread('Boya-2-cerca-de-taladro.jpg'));
%image(x),colormap(map), axis off, hold on

set(gcf,'name','Oceanografía Física');
img_ec=imread([cd '\img\ecvort_img3.bmp']);
axes(handles.axes6);
imshow(img_ec)                          % "image(img_ec),axis off"  es igual que
"imshow(img_ec)"
fondo1=imread([cd '\img\fondo1.bmp']);
axes(handles.axes7)
imshow(fondo1)

% Choose default command line output for modelo
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
initialize_gui(hObject, handles);                             % inicializa los controles e
indicadores

% UIWAIT makes modelowait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = modelo_OutputFcn(~, ~, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
```

```
varargout{1} = handles.output;



function im_val_Callback(hObject, ~, handles)
% hObject    handle to im_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of im_val as text
%        str2double(get(hObject,'String')) returns contents of im_val as a double

%Val=get(hObject,'String');      % Almacenar en Val el valor ingresado en formato string
%NewVal = str2double(Val);       % Transformar de string a formato double
%handles.im_val=NewVal;           % Almacenar NewVal en el identificador handles.im_val

im_val=str2double(get(hObject,'String'));       % almacenamos el string introducido en
formato double
if isnan(im_val)                            % si no se ha introducido un valor
   %set(hObject, 'String', 202);               % se preconfigura a un valor inicial de 202
   errordlg('Input must be a number','Error');
end
handles.data.im_val=im_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function im_val_CreateFcn(hObject, ~, handles)
% hObject    handle to im_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function jm_val_Callback(hObject, ~, handles)
% hObject    handle to jm_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of jm_val as text
%        str2double(get(hObject,'String')) returns contents of jm_val as a double

jm_val=str2double(get(hObject,'String'));
if isnan(jm_val)
   %set(hObject, 'String', 102);
   errordlg('Input must be a number','Error');
```

```
end
handles.data.jm_val=jm_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function jm_val_CreateFcn(hObject, ~, ~)
% hObject    handle to jm_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function ds_val_Callback(hObject, ~, handles)
% hObject    handle to ds_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ds_val as text
%        str2double(get(hObject,'String')) returns contents of ds_val as a double

ds_val=str2double(get(hObject,'String'));
if isnan(ds_val)
   %set(hObject, 'String', 0.05);
   errordlg('Input must be a number','Error');
end
handles.data.ds_val=ds_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function ds_val_CreateFcn(hObject, ~, ~)
% hObject    handle to ds_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function dt_val_Callback(hObject, ~, handles)
% hObject    handle to dt_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of dt_val as text
%        str2double(get(hObject,'String')) returns contents of dt_val as a double

dt_val=str2double(get(hObject,'String'));
if isnan(dt_val)
   %set(hObject, 'String', 0.05);
   errordlg('Input must be a number','Error');
end
handles.data.dt_val=dt_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function dt_val_CreateFcn(hObject, ~, ~)
% hObject    handle to dt_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function Ro_val_Callback(hObject, ~, handles)
% hObject    handle to Ro_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ro_val as text
%        str2double(get(hObject,'String')) returns contents of Ro_val as a double

Ro_val=str2double(get(hObject,'String'));
if isnan(Ro_val)
   %set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
handles.data.Ro_val=Ro_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function Ro_val_CreateFcn(hObject, ~, ~)
% hObject    handle to Ro_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function eps_val_Callback(hObject, ~, handles)
% hObject    handle to eps_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of eps_val as text
%        str2double(get(hObject,'String')) returns contents of eps_val as a double

eps_val=str2double(get(hObject,'String'));
if isnan(eps_val)
    %set(hObject, 'String', 0.3);
    errordlg('Input must be a number','Error');
end
handles.data.eps_val=eps_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function eps_val_CreateFcn(hObject, ~, ~)
% hObject    handle to eps_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Ah_val_Callback(hObject, ~, handles)
% hObject    handle to Ah_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Ah_val as text
%        str2double(get(hObject,'String')) returns contents of Ah_val as a double

Ah_val=str2double(get(hObject,'String'));
if isnan(Ah_val)
    %set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.data.Ah_val=Ah_val;
guidata(hObject,handles);       % Salvar datos de la aplicación
```

```
% --- Executes during object creation, after setting all properties.
function Ah_val_CreateFcn(hObject, ~, handles)
% hObject    handle to Ah_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function Bh_val_Callback(hObject, eventdata, handles)
% hObject    handle to Bh_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Bh_val as text
%        str2double(get(hObject,'String')) returns contents of Bh_val as a double

Bh_val=str2double(get(hObject,'String'));
if isnan(Bh_val)
   %set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
handles.data.Bh_val=Bh_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function Bh_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Bh_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function gamma_val_Callback(hObject, eventdata, handles)
% hObject    handle to gamma_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gamma_val as text
```

```
%       str2double(get(hObject,'String')) returns contents of gamma_val as a double

gamma_val=str2double(get(hObject,'String'));
if isnan(gamma_val)
   %set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
handles.data.gamma_val=gamma_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function gamma_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gamma_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function nst_val_Callback(hObject, eventdata, handles)
% hObject    handle to nst_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nst_val as text
%       str2double(get(hObject,'String')) returns contents of nst_val as a double

nst_val=str2double(get(hObject,'String'));
if isnan(nst_val)
   %set(hObject, 'String', 1);
   errordlg('Input must be a number','Error');
end
handles.data.nst_val=nst_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function nst_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nst_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end


function nend_val_Callback(hObject, eventdata, handles)
% hObject    handle to nend_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nend_val as text
%        str2double(get(hObject,'String')) returns contents of nend_val as a double

nend_val=str2double(get(hObject,'String'));
if isnan(nend_val)
   %set(hObject, 'String', 2000);
   errordlg('Input must be a number','Error');
end
handles.data.nend_val=nend_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function nend_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nend_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function nplot_val_Callback(hObject, eventdata, handles)
% hObject    handle to nplot_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nplot_val as text
%        str2double(get(hObject,'String')) returns contents of nplot_val as a double

nplot_val=str2double(get(hObject,'String'));
if isnan(nplot_val)
   %set(hObject, 'String', 100);
   errordlg('Input must be a number','Error');
end
handles.data.nplot_val=nplot_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function nplot_val_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to nplot_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function GYR_val_Callback(hObject, eventdata, handles)
% hObject    handle to GYR_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of GYR_val as text
%        str2double(get(hObject,'String')) returns contents of GYR_val as a double

GYR_val=str2double(get(hObject,'String'));
if isnan(GYR_val)
   %set(hObject, 'String', 1);
   errordlg('Input must be a number','Error');
end
handles.data.GYR_val=GYR_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function GYR_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GYR_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function HEM_val_Callback(hObject, eventdata, handles)
% hObject    handle to HEM_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of HEM_val as text
%        str2double(get(hObject,'String')) returns contents of HEM_val as a double

HEM_val=str2double(get(hObject,'String'));
```

```
if isnan(HEM_val)
   %set(hObject, 'String', -1);
   errordlg('Input must be a number','Error');
end
handles.data.HEM_val=HEM_val;
guidata(hObject,handles);       % Salvar datos de la aplicación

% --- Executes during object creation, after setting all properties.
function HEM_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to HEM_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function BFP_val_Callback(hObject, eventdata, handles)
% hObject    handle to BFP_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of BFP_val as text
%        str2double(get(hObject,'String')) returns contents of BFP_val as a double

BFP_val=str2double(get(hObject,'String'));
if isnan(BFP_val)
   %set(hObject, 'String', 1);
   errordlg('Input must be a number','Error');
end
handles.data.BFP_val=BFP_val;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function BFP_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BFP_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
% TITULO DE LA VENTANA PRINCIPAL
function text0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function im_lab_CreateFcn(hObject, eventdata, handles)
% hObject    handle to im_lab (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function jm_lab_CreateFcn(hObject, eventdata, handles)
% hObject    handle to jm_lab (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



% --- Executes during object creation, after setting all properties.
function ntime_val_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ntime_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function initialize_gui(fig_handle, handles)
% If the metricdata field is present and the reset flag is false, it means
% we are we are just re-initializing a GUI by calling it from the cmd line
% while it is up. So, bail out as we dont want to reset the data.

%if isfield(handles, 'metricdata') && ~isreset
%    return;
%end

% Se centra la figura en pantalla
set(handles.figure1,'units','pixels');
scrsz=get(0,'screensize');                              % te da el tamaño de la pantalla
pos_act=get(handles.figure1,'position');                % poosición actual de la figura
xr=scrsz(3)-pos_act(3);
xp=round(xr/2);
yr=scrsz(4)-pos_act(4);
yp=round(yr/2);
set(handles.figure1,'Position',[xp yp pos_act(3) pos_act(4)]);  % se centra la figura


% Se inicializan los valores de las variables de entrada
```

```
handles.data.im_val=202;
handles.data.jm_val=102;
handles.data.ds_val=0.05;
handles.data.dt_val=0.05;
handles.data.Ro_val=0;
handles.data.eps_val=0.3;
handles.data.Ah_val=0;
handles.data.Bh_val=0;
handles.data.gamma_val=0;
handles.data.nst_val=1;
handles.data.nend_val=2000;
handles.data.nplot_val=100;
handles.data.GYR_val=1;
handles.data.HEM_val=1;
handles.data.BFP_val=1;
%handles.data.Lx_val=1000;
%handles.data.Ly_val=500;

% Se muestran en pantalla los valores iniciales de las variables de entrada
set(handles.im_val, 'String', handles.data.im_val);
set(handles.jm_val, 'String', handles.data.jm_val);
set(handles.ds_val, 'String', handles.data.ds_val);
set(handles.dt_val, 'String', handles.data.dt_val);
set(handles.Ro_val, 'String', handles.data.Ro_val);
set(handles.eps_val, 'String', handles.data.eps_val);
set(handles.Ah_val, 'String', handles.data.Ah_val);
set(handles.Bh_val, 'String', handles.data.Bh_val);
set(handles.gamma_val, 'String', handles.data.gamma_val);
set(handles.nst_val, 'String', handles.data.nst_val);
set(handles.nend_val, 'String', handles.data.nend_val);
set(handles.nplot_val, 'String', handles.data.nplot_val);
set(handles.GYR_val, 'String', handles.data.GYR_val);
set(handles.HEM_val, 'String', handles.data.HEM_val);
set(handles.BFP_val, 'String', handles.data.BFP_val);
%set(handles.Lx_val, 'String', handles.data.Lx_val);
%set(handles.Ly_val, 'String', handles.data.Ly_val);

% Se inicializa el indicador 'Nº iteración'
set(handles.ntime_val, 'String', 0);

% Update handles structure
guidata(handles.figure1, handles);


% --- Executes on button press in Iniciar.
function Iniciar_Callback(hObject, eventdata, handles)
% hObject    handle to Iniciar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
% AQUI VA EL CÓDIGO QUE HACE LOS CÁLCULOS
```

```
% datestr(now,13)        % hh.mm.ss

data.im=handles.data.im_val;
data.jm=handles.data.jm_val;
data.ds=handles.data.ds_val;
data.dt=handles.data.dt_val;
data.Ro=handles.data.Ro_val;
data.eps=handles.data.eps_val;
data.Ah=handles.data.Ah_val;
data.Bh=handles.data.Bh_val;
data.gamma=handles.data.gamma_val;
data.nst=handles.data.nst_val;
data.nend=handles.data.nend_val;
data.nplot=handles.data.nplot_val;
data.GYR=handles.data.GYR_val;
data.HEM=handles.data.HEM_val;
data.BFP=handles.data.BFP_val;


% Se inicializan variables y se calcula el rotor de la tensión del viento,
% que se almacena en un .txt
[data_step,r,pa,pb,pc,psia,psib,psic,curlt,delpsi,delpsi4,fid1]...
   =QG_barotropM_comun_ini(data);


%Lx=handles.data.Lx_val;
%Ly=handles.data.Ly_val;
%nx=data.im-2;
%ny=data.jm-2;
%dx=Lx/(nx);
%dy=Ly/(ny);
%x=(0:nx-1)'*dx;
%y=(0:ny-1)'*dy;

TKE=nan(1,1);


for ntime=data.nst:data.nend

   [~,~,tke,p,psi,data_step,r,pa,pb,pc,psia,psib,...
   psic,delpsi,delpsi4]=...
      QG_barotropM_cada_iterac(data_step,r,pa,pb,pc,psia,psib,...
      psic,curlt,delpsi,delpsi4,fid1,ntime);

   set(handles.ntime_val, 'String', ntime); % -> muestra el nº iteracion
   drawnow

%    % SE HA DE TENER COMO SALIDA TODAS LAS VARIABLES Q SE VAN
ACTUALIZANDO,
%    % COMO data_step, r, pa, ... kp, ksf, kv, nplts
%    %
```

```
%    % p_c=pb(im/2,jm/2);       -> stream function in domain center
%    % psi_c=psib(im/2,jm/2);   -> vorticity in domain center
%    % tke: total kinetic energy
%    % p: stream function in current t
%    % psi: vorticity in current t

   %TKE(ntime,1)=tke;              % Ecinetica de cada iteración
   %psi_adim=p(2:end-1,2:end-1);
   %vor_adim=psi(2:end-1,2:end-1);


end

fclose(fid1);

guidata(hObject,handles);


% --- Executes on button press in Ayuda.
function Ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to Ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
hd=helpdlg({'im: nº puntos de la malla en dirección x';...
   'jm: nº puntos de la malla en dirección y';...
   'ds: resolución de la malla';...
   'dt: resolución temporal';...
   'Ro: nº de Rossby';...
   'eps: coef adim que representa la fricción del fondo';...
   'Ah: coef adim de la mezcla Laplaciana horizontal';...
   'Bh: coef adim de la mezcla biarmónica horizontal';...
   'gamma: coef del deslizamiento intermedio utilizado como condición de límite';...
   'nst: nº del paso temporal inicial';...
   'nend: nº del paso temporal final';...
   'nplot: frecuencia con la que se almacenan los datos';...
   'GYR: 1 para giro sencillo, 2 para giro doble';...
   'HEM: 1 para hemisferio norte, -1 para hemiferio sur';
   'BFP: 1 para plano Beta y 0 para plano F'},...
   'Definición de los parámetros del modelo');
% A LO MEJOR CONVIENE REPLAZARLO POR UN dialog



% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1
```

```
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
%
opc=questdlg('¿Desea salir del programa?','SALIR','Sí','No','No');
if strcmp(opc,'No')
   return;
end
delete(hObject);


% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

QG_barotropM_cada_iterac.m

```
function [p_c,psi_c,tke,p,psi,data_step,r,pa,pb,pc,psia,psib,...
   psic,delpsi,delpsi4]=...
   QG_barotropM_cada_iterac(data_step,r,pa,pb,pc,psia,psib,...
   psic,curlt,delpsi,delpsi4,fid1,ntime)
%.....................................................................
% This program solves the barotropic vorticity equation in non-dimensional
% form using finite differences.
% The model has incorporated the "partial" slipping boundary conditions
% The wind has been set for a single gyre problem
%.....................................................................
%
% model parameters
%
% im: # of points in the x-direction
% jm: # of points in the y-direction
% ds: grid-step
% dt: time-step
% Ro: Rossby number. It is a measure of the non-linearity of the flow.
% eps: non-dimensional coefficient representing bottom friction
% Ah: nondimensional coefficient of horizontal Laplacian mixing
% Bh: nondimensional coefficient of horizontal bi-harmonic mixing
% gamma: coefficient of "intermediate slipping" used as boundary condition

im=data_step.im;
jm=data_step.jm;
nplt=data_step.nplt;
ncrit=data_step.ncrit;
pcrit=data_step.pcrit;
dssqri=data_step.dssqri;
dssqri4=data_step.dssqri4;
rober=data_step.rober;
nplts=data_step.nplts;
c1=data_step.c1;
c2=data_step.c2;
c3=data_step.c3;
c4=data_step.c4;
c5=data_step.c5;
c6=data_step.c6;
c7=data_step.c7;
bc1=data_step.bc1;
bc2=data_step.bc2;
bc3=data_step.bc3;
imm1=data_step.imm1;
imm2=data_step.imm2;
jmm1=data_step.jmm1;
jmm2=data_step.jmm2;
%nplots=data_step.n_plots;
kp=data_step.kp;
alfa=data_step.alfa;
```

```
fxr=data_step.fxr;



% ----------------------------------------------------------------------
%
%              T i m e   I n t e g r a t i o n
%
%----------------------------------------------------------------------
   % Horizontal mixing
   delpsi=4*del2(psic).*dssqri;
   delpsi([1:2,imm1:im],:)=0;      delpsi(:,[1:2,jmm1:jm])=0;
   delpsi4=4*del2(delpsi).*dssqri;
   delpsi4([1:2,imm1:im],:)=0; delpsi4(:,[1:2,jmm1:jm])=0;

   for i=2:imm1
      for j=2:jmm1
         % Arakawa's jacobian
         jpp= ((pb(i+1,j)-pb(i-1,j)).*(psib(i,j+1)-psib(i,j-1))...
            -(pb(i,j+1)-pb(i,j-1)).*(psib(i+1,j)-psib(i-1,j))).*dssqri4;
         jxp= (psib(i,j+1).*(pb(i+1,j+1)-pb(i-1,j+1))...
            -psib(i,j-1).*(pb(i+1,j-1)-pb(i-1,j-1))...
            -psib(i+1,j).*(pb(i+1,j+1)-pb(i+1,j-1))...
            +psib(i-1,j).*(pb(i-1,j+1)-pb(i-1,j-1))).*dssqri4;
         jpx=(pb(i+1,j).*(psib(i+1,j+1)-psib(i+1,j-1))...
            -pb(i-1,j).*(psib(i-1,j+1)-psib(i-1,j-1))...
            -pb(i,j+1).*(psib(i+1,j+1)-psib(i-1,j+1))...
            +pb(i,j-1).*(psib(i+1,j-1)-psib(i-1,j-1))).*dssqri4;

         % Update vorticity
         psia(i,j)=c1.*psic(i,j)-c5.*(jpp+jxp+jpx)-c2.*(pb(i+1,j)...
            -pb(i-1,j))+c3.*curlt(i,j)-c4.*psic(i,j)...
            +c6.*delpsi(i,j) - c7.*delpsi4(i,j);
      end
   end
   clear i, clear j


   % Update stream function
   nrelax=0;
   nrelax=nrelax+1;
  while nrelax<=ncrit        % ncrit=4000
      % Solve the Laplacian for the stream function by over-relaxation
      for i=3:imm2
         for j=3:jmm2
            r(i,j)=((pa(i-1,j)+pa(i+1,j)+pa(i,j-1)+pa(i,j+1)...
               -4.*pa(i,j)).*dssqri-psia(i,j));
            pa(i,j)=pa(i,j)+alfa.*r(i,j).*fxr;
            % no se puede acortar xq 'r' depende del nuevo valor de 'pa'
         end
      end
```

```
      clear i, clear j

      % Check convergence
      n1=0;
      for i=3:imm2
         for j=3:jmm2
            rabs=sqrt(r(i,j).^2);   % -> QUE SENTIDO TIENE HACER LA RAIZ CUADRADA
DE ALGO ^2
            if rabs>pcrit          % pcrit=0.1
               n1=n1+1;
            end
         end
      end
      clear i, clear j
      % en fortran rabs es aprox =, pero n1 si que es =

      if n1~=0
         nrelax=nrelax+1;
      else
         break              % esto sucede en la iteración nrelax=169
      end
   end

   if nrelax<=ncrit        % si se cumple se continua la ejecución, si no
                           % finaliza la ejecución del programa
      % Set the boundary conditons on the stream function
      pa(1,2:jmm1)=bc1.*pa(3,2:jmm1);
      pa(2,2:jmm1)=0;
      pa(im,2:jmm1)=bc1.*pa(imm2,2:jmm1);
      pa(imm1,2:jmm1)=0;

      pa(2:imm1,1)=bc1.*pa(2:imm1,3);
      pa(2:imm1,2)=0;
      pa(2:imm1,jm)=bc1.*pa(2:imm1,jmm2);
      pa(2:imm1,jmm1)=0;


      % Diagonostic calculation of the vorticity on the walls
      psia(2,2:jmm1)=(pa(3,2:jmm1)+pa(1,2:jmm1)).*dssqri;
      psia(imm1,2:jmm1)=(pa(im,2:jmm1)+pa(imm2,2:jmm1)).*dssqri;

      psia(2:imm1,2)=(pa(2:imm1,3)+pa(2:imm1,1)).*dssqri;
      psia(2:imm1,jmm1)=(pa(2:imm1,jm)+pa(2:imm1,jmm2)).*dssqri;


      % Set the boundary conditons on the vorticity
      psia(1,2:jmm1)=bc3.*(bc2.*psia(3,2:jmm1)-4.*psia(2,2:jmm1)...
         +psia(2,3:jm)+psia(2,1:jmm2));
      psia(im,2:jmm1)=bc3.*(bc2.*psia(imm2,2:jmm1)-4.*psia(imm1,2:jmm1)...
         +psia(imm1,3:jm)+psia(imm1,1:jmm2));
```

```
psia(2:imm1,1)=bc3.*(bc2.*psia(2:imm1,3)-4.*psia(2:imm1,2)...
   +psia(3:im,2)+psia(1:imm2,2));
psia(2:imm1,jm)=bc3.*(bc2.*psia(2:imm1,jmm2)-4.*psia(2:imm1,jmm1)...
   +psia(3:im,jmm1)+psia(1:imm2,jmm1));


% Time smoothing the stream function using a Robert's filter

pb=pb+rober.*(pa-2.*pb+pc);
psib=psib+rober.*(psia-2.*psib+psic);


% Kinetic energy (entire domain)
tke=0;
for i=2:imm1
   for j=2:jmm1
     gradpsi=((pb(i+1,j)-pb(i-1,j)).^2+(pb(i,j+1)...
        -pb(i,j-1)).^2).*dssqri;
     tke=tke+0.5.*gradpsi;
   end
end
clear i, clear j

% Save data ======================
% write local stream function and vorticity (domain center) and
% total kinetic energy to file
fprintf(fid1, '%i \t %.12e \t %.12e \t %.6e \n\r', ...
   [ntime pb(im/2,jm/2) psib(im/2, jm/2) tke]);

p_c=pb(im/2,jm/2);
psi_c=psib(im/2,jm/2);

if ntime<nplts
   disp(['step number = ' num2str(ntime)])

else               % si el nº de iteración es = o mayor al nº de
   kp=kp+1;       % pasos en q se graba la salida
   num_file=num2str(kp,'%03u');

   disp(['step number = ' num2str(ntime) ...
      ' writing file = ' num_file])

   namefich1=['psi' num_file '.txt'];
   namefich2=['vor' num_file '.txt'];

   dlmwrite([cd '\out_tmp\' namefich1], pb',...
      'delimiter', '\t','precision','%1.3f');
   dlmwrite([cd '\out_tmp\' namefich2], psib',...
         'delimiter', '\t','precision','%1.3f');

   nplts=nplts+nplt;
```

```
    end

    p=pb;
    psi=psib;

    % update variables
    pc=pb;
    pb=pa;
    psic=psib;
    psib=psia;

    data_step.kp=kp;
    data_step.nplts=nplts;

else
    % Se finaliza la ejecución del programa
    disp(['Warning: The subroutine does not relax  ntime = ' ...
    num2str(ntime)])
    disp('Model run finished')
end
```

QG_barotropM_comun_ini.m

```
function [data_step,r,pa,pb,pc,psia,psib,psic,curlt,delpsi,delpsi4,fid1]...
   =QG_barotropM_comun_ini(data)
%.....................................................................
% This program solves the barotropic vorticity equation in non-dimensional
% form using finite differences.
% The model has incorporated the "partial" slipping boundary conditions
% The wind has been set for a single gyre problem
%.....................................................................
%
% model parameters
%
% im: # of points in the x-direction
% jm: # of points in the y-direction
% ds: grid-step
% dt: time-step
% Ro: Rossby number. It is a measure of the non-linearity of the flow.
% eps: non-dimensional coefficient representing bottom friction
% Ah: nondimensional coefficient of horizontal Laplacian mixing
% Bh: nondimensional coefficient of horizontal bi-harmonic mixing
% gamma: coefficient of "intermediate slipping" used as boundary condition

im=data.im;
jm=data.jm;
ds=data.ds;
dt=data.dt;
Ro=data.Ro;
eps=data.eps;
Ah=data.Ah;
Bh=data.Bh;
gamma=data.gamma;
nst=data.nst;
nend=data.nend;
nplt=data.nplot;
GYR=data.GYR;
HEM=data.HEM;
BFP=data.BFP;
ncrit=4000;
pcrit=0.1;

% Model parameters are load ----------------------------------
%im=202;     % number of grid points in the zonal direction
%jm=102;     % number of grid points in the meridional direction
%ds=0.05;    % grid step
%dt=0.05;    % time step
%Ro=0.0;     % Rossby number (measures non-linearity of the flow)
%eps=0.3;    % non-dimensional coefficient representing bottom friction
%Ah=0.0;     % non-dimensional coeff. of horizontal Laplacian mixing
%Bh=0.0;     % non-dimensional coeff. of horizontal bi-harmonic mixing
%gamma=0.0;  % coeff. of "intermediate slipping" used as boundary cond.
```

```
%nst=1;      % start time step number
%nend=2000;  % end time step number
%nplt=100;   % frequency (time steps) for saving output
%ncrit=4000; % number of steps allowed to do the relaxation (sub. helm)
%pcrit=0.1;  % criterium to stop the relaxation
%BFP=1;      % Beta (BFP=1) or F plane (BFP=0)
%GYR=1;      % Simple Gyre (GYR=1) or Double Gyre (GYR=2)
%HEM=-1;     % North Hemisphere Gyre (HEM=1) or South Hemisphere Gyre (HEM=-1)
%QG_param

% Arrays ----------------------------------------------------
% p: stream function
% psi: Laplacian of the stream function??? o vorticidad
% curlt: wind stress curl
% delpsi: delta 4th of the stream function. Used for horizontal mixing
%
% Note: the subscript a,b,c denotes time steps t+1,t,t-1 respectively.

% General constants -----------------------------------------
ele=2*pi/(jm-1);
kx=pi/(im-1);
dssqr=ds.^2;
dssqri=1/dssqr;
dssqri4=dssqri/4;
epsdt=eps*dt;
epsdti=1/(1+epsdt);
rober=1e-3;
nplts=nplt;
c1=epsdti;
c2=BFP.*(dt./ds).*epsdti;
c3=2.*dt.*epsdti;
c4=epsdt.*epsdti;
c5=2.*dt.*epsdti.*Ro./3.0;
c6=2.*dt.*Ah.*epsdti;
c7=2.*dt.*Bh.*epsdti;

bc1=-(1.-gamma.*ds.*0.5)./(1.+gamma.*ds.*0.5);
bc2=1.-gamma.*ds.*0.5;
bc3=- 1./(1. + gamma.*ds.*0.5);

imm1=im-1;
imm2=im-2;
jmm1=jm-1;
jmm2=jm-2;


% Number of time steps to be saved ---------------------------------
n_plots=fix((nend-nst)/nplt)+1;       % -> en realidad se ha redondear al más cercano
                    % (round), pero no me coincide con fortran
disp(['Time steps to be saved = ' num2str(n_plots)])
kp=0;
```

```
% Over-relaxation constants ----------------------------------------------
% ncrit: # of steps allowed to do the relaxation in subroutine helm
% pcrit: criterium to stop the relaxation
% alfa: constant used for the sequential relaxation
% fxr:  constant used for the sequential relaxation
const1= 1./(imm2-2).^2;
const2= 1./(jmm2-2.).^2;
alfa=2.-4.442883.*sqrt(const1+const2);
fxr=dssqr./4.0;


% initialize all arrays ---------------------------------------------------
r=zeros(im,jm);
pa=zeros(im,jm);        % Función corriente
pb=zeros(im,jm);
pc=zeros(im,jm);
psia=zeros(im,jm);      % Laplaciano de la función corriente
psib=zeros(im,jm);
psic=zeros(im,jm);
curlt=zeros(im,jm);     % Fuerza del viento
delpsi=zeros(im,jm);
delpsi4=zeros(im,jm);   % 4ª delta de la función corriente


% Wind stress curl --------------------------------------------------------
jm=102; im=202;GYR=1;HEM=-1;ele=2*pi/(jm-1);
J=repmat((0:jm-1),im,1);
if GYR==1               % Single gyre
   curlt=HEM.*sin(ele.*0.5.*J);
elseif GYR==2           % Double gyre
   I2=repmat((0:im-1)',1,jm);
   curlt=-sin(ele.*0.5.*J).*sin(kx.*I);
   clear I
else
   disp('Wind Gyre type not defined')
   return               % finaliza la ejecución de la actual función si no
end                     % esta definido el tipo de viento
clear J

dlmwrite([cd '\out_tmp\QG_wind_stress.txt'], curlt',...
    'delimiter', '\t','precision','%1.3f');

fid1=fopen([cd '\out_tmp\QG_diag.txt'],'w');

data_step.im=im;
data_step.jm=jm;
data_step.nplt=nplt;
data_step.ncrit=ncrit;
data_step.pcrit=pcrit;
```

```
data_step.dssqri=dssqri;
data_step.dssqri4=dssqri4;
data_step.rober=rober;
data_step.nplts=nplts;
data_step.c1=c1;
data_step.c2=c2;
data_step.c3=c3;
data_step.c4=c4;
data_step.c5=c5;
data_step.c6=c6;
data_step.c7=c7;
data_step.bc1=bc1;
data_step.bc2=bc2;
data_step.bc3=bc3;
data_step.imm1=imm1;
data_step.imm2=imm2;
data_step.jmm1=jmm1;
data_step.jmm2=jmm2;
%data_step.n_plots=n_plots;  % nºde veces q se representan y graban datos
data_step.kp=kp;
data_step.alfa=alfa;
data_step.fxr=fxr;

end
```