PRÁCTICA 3

Se trata de un proyecto de programación individual en el que explicáis los programas que hemos realizado en clase. Obviamente, deseo autoría por vuestra parte, con lo cual os pediré que desarrolléis los siguientes puntos en vuestra memoria:

1. Que expliquéis el problema directo y el problema inverso.
2. Que programéis otros tipos de anomalía de densidad, a saber:
    a. Densidad Gaussiana

$$\rho(x;A,B,\mu,\sigma) = A + \frac{B}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

    b. Densidad constante a trozos

$$\rho(x;\rho_k,x_j) = \begin{cases} \rho_0 & si \quad x \in [x_0,x_1] \\ \rho_1 & si \quad x \in [x_1,x_2] \\ \rho_2 & si \quad x \in [x_2,x_3] \end{cases}$$

Se pide resolver el problema directo y e inverso para ambos tipos de anomalías, y que analizar la estructura de la matriz del sistema, la sensibilidad del método a la profundidad del cuerpo denso, al contraste de densidad y a la situación del mismo en cuanto a la campaña de adquisición de datos. Analizar también el efecto del ruido en la inversión.

```
%
%---------------PROGRAM TO SOLVE THE GRAVIMETRIC PROBLEM----
%
%                      INVERSE GRAVIMETRIC PROBLEM 1D
%            BURIED BODY LOCATED IN [xl xr] AT A CONSTANT DEPTH D
%                           Juan Luis Fernández-Martínez
%                              Zuliam Fernández-Muñiz
%-------------------------------------------------------------------
-----
%
clear all
% close all
clc
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% FIRST STEP: Generation data patterns with different density
functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%
G = 6.67e-11; % Gravity universal constant
n = 5000; % number of parameters
si = -10; % Initial coordinate for the measure points
sf = 10; % Final coordinate for the measure points
ndata=1250;
sdato = linspace(si,sf,ndata); % intervalo de los datos
D = 2; % Depth of the buried body
xl = -3; % Initial coordinate the dense body
xr = 6; % Final coordinate the dense body
xd = linspace(xl,xr-0.001,n); % intervalo de los parámetros
val = 2; % Constant density value
percentage = 0; % Noise percentage to perturb the observed data
%
% Generated data from different density model
%
x = linspace(xl,xr,n);
[gobs] = gravity_cte(x,sdato,D,val,xl,xr,percentage);
%
%  Density function
%
dval = density_cte(x,xl,xr,val);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% SECOND STEP: Calculation of matrix system
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%
npixels = n;
matrix = matrix_pixel(sdato,xl,xr,D,npixels);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% THIRD STEP: Solution of the inverse problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

```
%
coef_pinv = pinv(matrix,1e-4)*gobs.';

%  Inverse problem solution
%
modelinv_pinv = pixel_component(xd,coef_pinv,xl,xr);
%
% Calculo de los datos a partir del modelo encontrado
%
gobs_pinv = matrix*coef_pinv;
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% FOURTH STEP: Printing the results
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%
%  Gravity function
%
figure
plot(sdato,gobs_pinv,'k.-')
hold on
plot(sdato,gobs,'r.-')
xlabel('x (m)');
ylabel('gravity (m/s^2)');
legend('predicted gravity','gravity anomaly')
%
%  Density function
%
figure
hold on
plot(xd,modelinv_pinv,'k.-');
plot(xd,dval,'r-')
xlabel('x (m)');
ylabel('density (kg/m^3)');
legend('predicted density','density anomaly')
axis ([xl,xr,0,1.2*max(modelinv_pinv)])
```

% <u>FUNCIÓN GRAVITY CTE:</u>

```
function [gobs]=gravity_cte(x,sdato,D,val,xinicio,xfinal,percentage)
%
gobs=zeros(1,length(sdato));
for k=1:length(sdato)
    xdato=sdato(k);
    gobs(k)=quad(@(x) (D*val./((D^2+(x-
xdato).^2)).^(3/2)),xinicio,xfinal);
end
% Total noise added to the data
ruido=randn(1,length(gobs))*mean(gobs)*percentage;
% Modified data with the added noise
gobs=gobs+ruido;
```

% FUNCIÓN DENSITY CTE

```
function [dval]=density_cte(x,xl,xr,val)
% val : value of the constant density
dval=val*(x>=xl & x<=xr);
```

% FUNCIÓN MATRIX PIXEL

```
function [sens]=matrix_pixel(sdato,a,b,D,npixels)
ndata=length(sdato);
sens=zeros(ndata,npixels);
for i=1:ndata
    xdato=sdato(i);
    % Proyection over constant piecewise functions
    for k = 1:npixels
        ak=a+(k-1)*(b-a)/npixels;
        bk=a+k*(b-a)/npixels;
        % Integral value of the product of the kernel by the
corresponding basis function
        sens(i,k)=int_nucleo(bk,xdato,D)-int_nucleo(ak,xdato,D);
    end
end
```

% FUNCIÓN INTEGRAR NÚCLEO

```
function [pval]=int_nucleo(x,s,D);
% función primitiva exacta del nucleo
num=(1/D)*(x-s);
den=sqrt(D^2+(x-s).^2);
pval=num./den; %Valor de la primitiva del núcleo
```

% FUNCIÓN PIXEL COMPONENT

```
function [fpixel]=pixel_component(x,coef,a,b)
% Number of point where the aproximate function is calculated
nrepresenta = length(x);
% Number of piecewise functions which value is 1
npixels = length(coef);
% Inicializacion de la funcion aproximada
fpixel=zeros(1,nrepresenta);
for i=1:nrepresenta
    xi=x(i);
    for k= 1:npixels
        % Coordenada del principio del soporte de la funcion k-esima
de valor 1
        ak=a+(k-1)*(b-a)/npixels;
        % Coordenada del fin del soporte de la funcion k-esima de
valor 1
        bk=a+k*(b-a)/npixels;
        % Definicion de la funcion k-esima en el intervalo de
definicion correspondiente
        aux(k)=1*(xi>=ak & xi<bk);
        % Valor de la funcion aproximada
        fpixel(i) = fpixel(i)+coef(k)*aux(k);
    end
end
```