Table of Contents

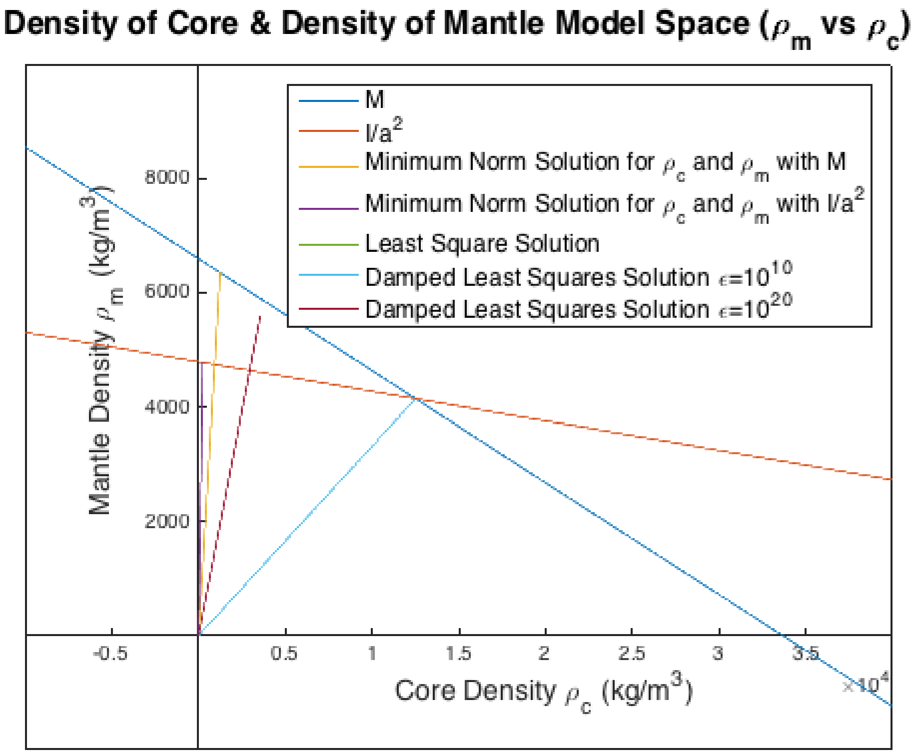
2.a)-d) Generating plots for M and I/a^2

```
close all;
clc; clear all;

% Defining variables
c=3485000; % radius of core
a=6371000; % radius of Earth
M=5.972*10^24; % mass of the Earth
% I=(2/5)*M*(a^2); % moment of inertia of a Sphere (too approximate)
I=8.008*10^37; % Courtesy of Lambeck (1980) The Earth's Variable
 Rotation (book)

%Plotting the model space for M

%pm=0:
pc=3*(M)/(4*pi*c^3);

%pc=0:
pm=3*(M)/(4*pi*((a^3)-(c^3)));

m1=pm/(-1*pc); % calculating the slope for the line
pcx=[-10000:1:40000];
pmplot=m1*(pcx)+pm; % y=mx + b straight line formula

figure;
plot(pcx,pmplot); % plot the straight line (model space of pm vs pc)
xlabel('Core Density \rho_c (kg/m^3)','FontSize',14);
ylabel('Mantle Density \rho_m (kg/m^3)','FontSize',14);
set(gca,'XAxisLocation','origin','YAxisLocation','origin');
title('Density of Core & Density of Mantle Model Space (\rho_m vs
 \rho_c)','FontSize',16);
hold on;

%%Plotting the model space for I/a^2

%when pm=0;
pc2=(15*I)/(8*pi*(c^5));
```

```
%when pc=0;
pm2=(15*I)/(8*pi*((a^5)-(c^5)));

m2=pm2/(-1*pc2); % calculating the slope for the line
pmplot2=m2*(pcx)+pm2; % y=mx + b straight line formula
plot(pcx,pmplot2); % plot the straight line (model space of pm vs pc)
hold on;

% 2. b) Generating solutions for pc and pm minimum norm appraoch

% Calculating and plotting result of finding pc and pm using mass of
 the Earth (M)
G1=[(4/3)*pi*c^3 (4/3)*pi*((a^3)-(c^3))];
pcpm1=(G1')*inv((G1)*(G1'))*M
pcx2=[0:1:1246];
m3=pcpm1(2)/(pcpm1(1));
q1=m3*(pcx2);
plot(pcx2,q1);
hold on;

% Calculating and plotting result of finding pc and pm using Inertia
 (I)
G2=(4*pi/3)*[(2*c^5)/(5*a^2) (2/5)*((a^3)-((c^5)/(a^2)))];
pcpm2=(G2')*inv((G2)*(G2'))*I/(a^2)
pcx3=[0:1:246];
m4=pcpm2(2)/pcpm2(1);
q2=m4*(pcx3);
plot(pcx3,q2);
hold on;

% 2. c) Calculting least squares solution for the combined problem
 (using
% both M and I/a^2)

d=[M; (I/(a^2))];
G3=[G1; G2];
pcpm3=inv((G3')*G3)*(G3')*d
pcx4=[0:1:12512];
m5=pcpm3(2)/pcpm3(1);
q3=m5*(pcx4);
plot(pcx4,q3);
hold on;

% 2. d) % damped least squares with white noise
epsilon1=[10^10 0;0 10^10]
pcpm4=(inv(G3'*G3+(epsilon1.^2)))*(G3')*d
pcx5=[0:1:12512];
m6=pcpm4(2)/pcpm4(1);
q4=m6*(pcx4);
plot(pcx5,q4);
hold on;

epsilon2=[10^20 0;0 10^20]
pcpm5=(inv(G3'*G3+(epsilon2.^2)))*(G3')*d
```

```
pcx6=[0:1:3575];
m7=pcpm5(2)/pcpm5(1);
q5=m7*(pcx6);
plot(pcx6,q5);
legend=legend('M','I/a^2','Minimum Norm Solution for \rho_c and
 \rho_m with M','Minimum Norm Solution for \rho_c and \rho_m with
 I/a^2','Least Square Solution', 'Damped Least Squares Solution
 \epsilon=10^1^0','Damped Least Squares Solution \epsilon=10^2^0');
set(legend,'Fontsize',12')


pcpm1 =

   1.0e+03 *

    1.2426
    6.3491


pcpm2 =

   1.0e+03 *

    0.2459
    4.7752


pcpm3 =

   1.0e+04 *

    1.2512
    0.4144


epsilon1 =

   1.0e+10 *

    1.0000         0
         0    1.0000


pcpm4 =

   1.0e+04 *

    1.2512
    0.4144


epsilon2 =

   1.0e+20 *
```

```
    1.0000         0
         0    1.0000


pcpm5 =

   1.0e+03 *

    3.5749
    5.6150
```



3. a) Writing out G, d, G', G'G, and GG'

```
close all; clear all; clc;
G=[1 1 0; 0 0 1/2; 0 0 -1]
d=[1;2;1]
G'
A=G'*G % defining A as the transpose of G times G
B=G*G' % defining B as G times the transpose of G
```

```
G =

    1.0000    1.0000         0
```

```
         0         0    0.5000
         0         0   -1.0000


d =

     1
     2
     1


ans =

    1.0000         0         0
    1.0000         0         0
         0    0.5000   -1.0000


A =

    1.0000    1.0000         0
    1.0000    1.0000         0
         0         0    1.2500


B =

    2.0000         0         0
         0    0.2500   -0.5000
         0   -0.5000    1.0000
```

3. b) Invertibility of G'G and GG'

```
detA=det(A) % taking the determinant of A
detB=det(B) % taking the determinant of B


detA =

     0


detB =

     0
```

3. c) Solving for unknown parameters m with the method of damped least squares

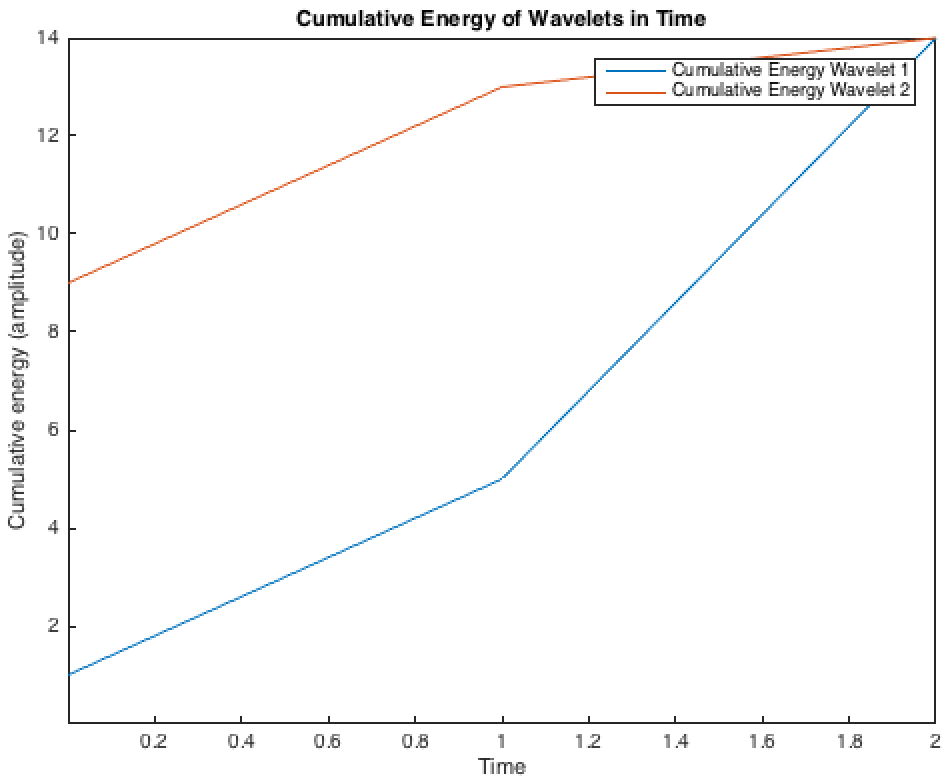
```
ep1=0.01 % defining epsilon 1
```

```
ep2=0.1 % defining epsilon 2
q1=[ep1^2 0 0;0 ep1^2 0; 0 0 ep1^2] % epsilon squared times the
 identity matrix
q2=[ep2^2 0 0;0 ep2^2 0; 0 0 ep2^2] % epsilon squared times the
 identity matrix

% method of damped least squares
m1=(inv(A+q1))*(G')*d % solving for unknown parameters m with
 epsilon=0.01
m2=(inv(A+q2))*(G')*d % solving for unknown parameters m with
 epsilon=0.1


ep1 =

    0.0100


ep2 =

    0.1000


q1 =

   1.0e-04 *

    1.0000         0         0
         0    1.0000         0
         0         0    1.0000


q2 =

    0.0100         0         0
         0    0.0100         0
         0         0    0.0100


m1 =

    0.5000
    0.5000
         0


m2 =

    0.4975
    0.4975
         0
```

4. a) Cumulative energy of wavelets plots

```
%cumulative energy wavelets
w1=[1^2 (1^2)+((-2)^2) (1^2)+((-2)^2)+3^2]
w2=[3^2 (3^2)+((-2)^2) (3^2)+((-2)^2)+1^2]
t=[0:1:2]
figure;
plot(t,w1,t,w2)
title('Cumulative Energy of Wavelets in Time')
set(gca,'XAxisLocation','origin','YAxisLocation','origin');
legend('Cumulative Energy Wavelet 1','Cumulative Energy Wavelet 2')
xlabel('Time')
ylabel('Cumulative energy (amplitude)')
```

```
w1 =

     1     5    14


w2 =

     9    13    14


t =

     0     1     2
```



4. b) calculating three-element wiener inverse filters

```
G1=[1 0 0;-2 1 0;3 -2 1;0 3 -2; 0 0 3] % defining the matrix G1 for
 wavelet 1
G2=[3 0 0;-2 3 0;1 -2 3;0 1 -2; 0 0 1] % defining the matrix G2 for
 wavelet 2

d1=[1;0;0;0;0] % defining desired output d1
d2=[0;1;0;0;0] % defining desired output d2

% calculting a filter using a least squares solution for wavelet 1
 with both d1 and d2
f1= (inv(G1'*G1))*(G1'*d1)
f2= (inv(G1'*G1))*(G1'*d2)

% calculating a filter using a least squares solution for wavelet 2
 with
% both d1 and d2
f3= (inv(G2'*G2))*(G2'*d1)
f4= (inv(G2'*G2))*(G2'*d2)
```

G1 =

```
     1      0      0
    -2      1      0
     3     -2      1
     0      3     -2
     0      0      3


G2 =

     3      0      0
    -2      3      0
     1     -2      3
     0      1     -2
     0      0      1


d1 =

     1
     0
     0
     0
     0


d2 =

     0
     1
     0
     0
     0


f1 =

    0.1091
    0.0727
    0.0182


f2 =

   -0.1455
    0.0091
    0.0364


f3 =

    0.3273
    0.2182
    0.0545
```

```
f4 =

    0.0000
    0.3182
    0.1818
```

4. c) Apply inverse operators from (b) to wavelet 1 and wavelet 2

```
% applying the filters to the G1 matrix for wavelet 1, trying to get
 back
% desired outputs d1 and d2
r1= G1*f1 % desired output is d1
r2= G1*f2 % desired output is d2

% applying the filters to the G2 matrix for wavelet 1, trying to get
 back
% desired output d2
r3=G2*f3 % desired output is d1
r4=G2*f4 % desired output is d2


r1 =

    0.1091
   -0.1455
    0.2000
    0.1818
    0.0545


r2 =

   -0.1455
    0.3000
   -0.4182
   -0.0455
    0.1091


r3 =

    0.9818
    0.0000
    0.0545
    0.1091
    0.0545


r4 =
```

```
    0.0000
    0.9545
   -0.0909
   -0.0455
    0.1818
```

4. d) damped least squares solutions (adding white noise)

```
ep1=0.01; % defining epsilon 1
ep2=0.1; % defining epsilon 2
q1=[ep1^2 0 0;0 ep1^2 0; 0 0 ep1^2;]; % epsilon squared times the
 identity matrix
q2=[ep2^2 0 0;0 ep2^2 0; 0 0 ep2^2]; % epsilon squared times the
 identity matrix

% calculting filters using a damped least squares solution for wavelet
 1
% with white noise
b1= (inv(G1'*G1+q1))*(G1'*d1) % epsilon=0.01 with d1=[1;0;0;0;0];
b2= (inv(G1'*G1+q2))*(G1'*d1) % epsilon=0.1 with d1=[1;0;0;0;0];
b3= (inv(G1'*G1+q1))*(G1'*d2) % epsilon=0.01 with d2=[0;1;0;0;0];
b4= (inv(G1'*G1+q2))*(G1'*d2) % epsilon=0.1 with d2=[0;1;0;0;0];

% calculating filters using a damped least squares solution for
 wavelet 2
% with white noise
b5= (inv(G2'*G2+q1))*(G2'*d1) % epsilon=0.01 with d1=[1;0;0;0;0];
b6= (inv(G2'*G2+q2))*(G2'*d1) % epsilon=0.1 with d1=[1;0;0;0;0];
b7= (inv(G2'*G2+q1))*(G2'*d2) % epsilon=0.01 with d2=[0;1;0;0;0];
b8= (inv(G2'*G2+q2))*(G2'*d2) % epsilon=0.1 with d2=[0;1;0;0;0];

% applying the new filters (with noise) to the original wavelets

% applying filters to wavelet 1
des1=G1*b1 % Aiming to get d1=[1;0;0;0;0]
des2=G1*b2 % Aiming to get d1=[1;0;0;0;0]
des3=G1*b3 % Aiming to get d2=[0;1;0;0;0]
des4=G1*b4 % Aiming to get d2=[0;1;0;0;0]

% applying filters to wavelet 2
des5=G2*b5 % Aiming to get d1=[1;0;0;0;0]
des6=G2*b6 % Aiming to get d1=[1;0;0;0;0]
des7=G2*b7 % Aiming to get d2=[0;1;0;0;0]
des8=G2*b8 % Aiming to get d2=[0;1;0;0;0]
```

```
b1 =

    0.1091
```

```
    0.0727
    0.0182


b2 =

    0.1089
    0.0725
    0.0181


b3 =

   -0.1455
    0.0091
    0.0364


b4 =

   -0.1453
    0.0092
    0.0363


b5 =

    0.3273
    0.2182
    0.0545


b6 =

    0.3267
    0.2176
    0.0543


b7 =

   -0.0000
    0.3182
    0.1818


b8 =

   -0.0003
    0.3176
    0.1814


des1 =
```

```
    0.1091
   -0.1455
    0.2000
    0.1818
    0.0545


des2 =

    0.1089
   -0.1453
    0.1998
    0.1814
    0.0543


des3 =

   -0.1455
    0.3000
   -0.4182
   -0.0455
    0.1091


des4 =

   -0.1453
    0.2998
   -0.4179
   -0.0452
    0.1090


des5 =

    0.9818
   -0.0000
    0.0545
    0.1091
    0.0545


des6 =

    0.9802
   -0.0008
    0.0544
    0.1090
    0.0543


des7 =
```

```
   -0.0000
    0.9545
   -0.0909
   -0.0455
    0.1818


des8 =

   -0.0008
    0.9532
   -0.0912
   -0.0452
    0.1814
```

4. e) Comment on the effectiveness of the filters as applied to the wavelets

```
% For wavelet 1 (as defined by matrix G1 in 4.b), the filters are not
% effective, because the wavelet is not minimum phase! The spiking
% deconvolution method that was used assumes a minimum phase wavelet,
 and
% because it the wavelet is not minimum phase, the filter does not
 produce the desired outputs. None of the results
% of convolving the filter with the wavelet were close to the desired
 outputs when the
% first wavelet (G1) was used. The outputs were never close to
% d1=[1;0;0;0;0] or d2=[0;1;0;0;0].

% However, wavelet 2 (G2) is minimum phase. The filters produced very
 good
% results, coming close to d1=[1;0;0;0;0] and d2=[0;1;0;0;0] (eg.
 des8). This should
% be the case for spiking deconvolution, as we assume a minimum phase
% wavelet.
```

Published with MATLAB® R2015b