Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
CE-3102 Análisis Numérico para Ingeniería
Profesor: Dr. Pablo Alvarado Moya
I Semestre, 2018
Examen Final
Nombre: Jairo Ortega Calderón
Carné: 2014043224

Problema 1:

1.1 Grafique los puntos bidimensionales utilizando GNU/Octave.

Para realizar este punto hay que usar la función plot y definir como los ejes x,y las filas 1 y 2 de la matriz X respectivamente, como se ve la figura 1.1. El resultado puede observarse en la figura 1.2.

```
20  ######################################################
21  ## Problema 1.1                                     ##
22  ## Grafique los puntos bidimensionales              ##
23  ######################################################
24  figure(1);
25  plot(X(1,1:end),X(2,1:end),'.' );
26  xlabel("X")
27  ylabel("Y")
28  grid on;
29  hold off;
```

Figura 1.1. Código del punto 1.1.

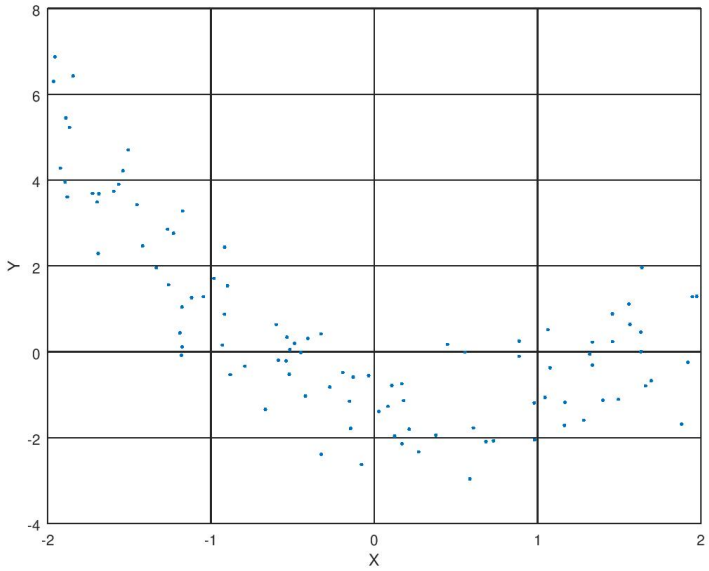

Figura 1.2. Resultado de graficar el punto 1.1

1.2 Implemente la función de error. Debe seguir la interfaz especificada en el archivo brindado.

Para realizar este punto se implementa la función de error que brinda la descripción del problema. La sumatoria se realiza con el while poniendo como limite la cantidad de la matriz X.

```
31 ##################################################
32 ## Problema 1.2                                  ##
33 ## Implemente la función de error                ##
34 ##################################################
35 function val=f(abc,X)
36   ## abc: vector columna [a,b,c]' con los parámetros de la función cuadrática
37   ## X:   datos para evaluar la función, un dato por columna
38
39   ## Ponga su código aquí:
40   iSum=1;
41   tempVal=0;
42   mSum=columns(X);
43   while(iSum<=mSum)
44     tempVal+=(X(2,iSum) - (abc(1)*(X(1,iSum)^2) + abc(2)*X(1,iSum) + abc(3)))^2;
45     iSum+=1;
46   endwhile
47   val=tempVal;
48
49 endfunction
```

Figura 1.3. Código de función de error.

1.3. Utilizando las técnicas de diferenciación numérica vistas en clase, calcule el gradiente.

Para este punto decidí utilizar diferenciación centrada, debido a que como fue visto en clase y demostrado en una tarea está diferenciación es la que se acerca mas al dato “real”. Esta diferenciación utiliza la diferenciación adelante y atrás, en el proyecto se incluyen los archivos mas no se muestran en este documento para no agregar mas imagenes, pero basícamente los métodos consisten en sacar las derivadas de la función y para obtener la centrada se resta la diferenciación de adelante menos la de atrás.

```
52 ##################################################
53 ## Problema 1.3                                  ##
54 ## Implemente el gradiente de la función de error ##
55 ##################################################
56 function val=gf(abc,X)
57   ## abc: vector columna [a,b,c]' con los parámetros de la función cuadrática
58   ## X:   datos para evaluar la función, un dato por columna
59
60   ## Use diferenciación NUMERICA para calcular el gradiente de f:
61   iSum=1;
62   tempVal=[0,0,0];
63   mSum=columns(X);
64   global h;
65   while(iSum<=mSum)
66     tempVal+=diferenciaCentrada(abc,X(1,iSum), X(2,iSum), h);
67     iSum+=1;
68
69   endwhile
70   val=tempVal';
71
72 endfunction
```

Figura 1.4. Código del punto 1.3

1.4 Implemente un ciclo para encontrar, utilizando la regla Δ (también conocida como descenso de gradiente) los valores de a, b y c que minimizan f (a, b, c).

Para realizar este punto primero se utiliza la variable global llamada h, para modificar los pasos que se utilizan en el gradiente, este valor se divide a la mitad de cada iteración a partir del lambda. La ejecución se detiene cuando el valor absoluto del error porcentual aproximado sea menor a la tolerancia, ya que este fue el método escogido por mi y se calcula evaluando la función con el valor actual de a,b,c y el valor anterior. Finalmente el valor de lambda y la tolerancia fue elegido a partir de prueba y error. El código se puede ver en la siguiente figura.

```
  ## Ponga su código aquí:
  errorAprox=[];
  ABCAprox=[];
  ea=errorPorcentualAproximado(f(abc0', X),0);
  aproxAct=abc0';
  global h;
  h=lambda;
  n=1;
  while(abs(ea)>tol)
  ABCAprox=[ABCAprox;aproxAct];
  errorAprox=[errorAprox;abs(ea)];
  aproxAct=ABCAprox(rows(ABCAprox),1:end)-lambda*gf(ABCAprox(rows(ABCAprox),1:end),X)';
  ea=errorPorcentualAproximado(f(aproxAct,X),f(ABCAprox(rows(ABCAprox),1:end),X));
  h=h*0.5;
  n=n+1;
  endwhile
  err=errorAprox;
  ABC=ABCAprox;
endfunction

## Llame al optimizador con la interfaz anterior

lambda=0.001;  # Ajuste esto
tol=0.00000001; # Ajuste esto
[ABC,err]=optimice(@f,@gf,X,lambda,tol,[0,1,0]');
```

Figura 1.5. Código del punto 1.4

1.5 Encuentre cuál es el conjunto de parámetros a, b y c óptimo.

Los valores óptimos son los obtenidos como ultimo paso del punto 1.4, por lo tanto solo se escogen de la última fila de ABC. Los valores obtenidos son a = 1.1135270, b = -1.1009378, c = -1.1261925. En la siguiente imagen se observa la obtención de esos valores..

```
##################################################
## Problema 1.5                                  ##
## Imprima el conjunto óptimo de parámetros      ##
##################################################
ABCOptimo=ABC(rows(ABC),1:end);
a=ABCOptimo(1)
b=ABCOptimo(2)
c=ABCOptimo(3)
```

Figura 1.6. Código de a,b y c óptimo.

1.6. Grafique el error en función del número de iteración en su proceso de optimización. Etiquete los ejes.

Para hacer este punto solo se toma los datos de error obtenidos en el punto 1.4 y se gráfican según su orden como se observa en el código de la siguiente imagen y el resultado se encuentra en la figura 1.8.

```
130 ######################################################
131 ## Problema 1.6                                     ##
132 ## Muestre el error en función de las iteraciones ##
133 ######################################################
134
135 figure(2);
136 plot(1:rows(err),err(1:end,1),'.' );
137 title('Error Porcentual Aproximado vs Número de iteración')
138 xlabel('Número de iteración')
139 ylabel('Error Porcentual Aproximado')
140 grid on;
141 hold off;
```

Figura 1.7. Código del punto 1.6

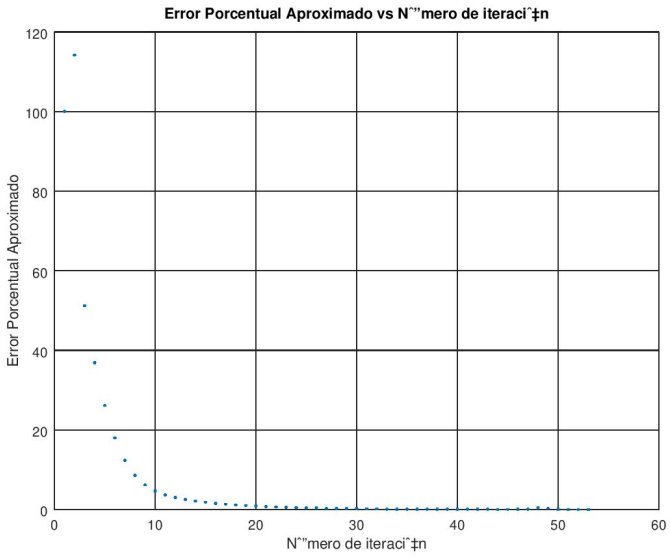

Figura 1.8. Gráfica de error porcentual aproximado.

1.7. Grafique en el invervalo de x ∈ [−2, 2] sobre la figura del punto 1.1 lo siguiente:

1. **la curva asociada al punto de partida en color negro**
2. **las curvas asociadas a cada paso intermedio del proceso de optimización en color cian**
3. **la curva asociada al conjunto óptimo de parámetros en color rojo.**

Como se puede observar en la siguiente figura el resultado obtenido es similar al que se brinda en la imagen del enunciado y para obtenerlo solamente se grafican con la función plot de los datos solicitados.

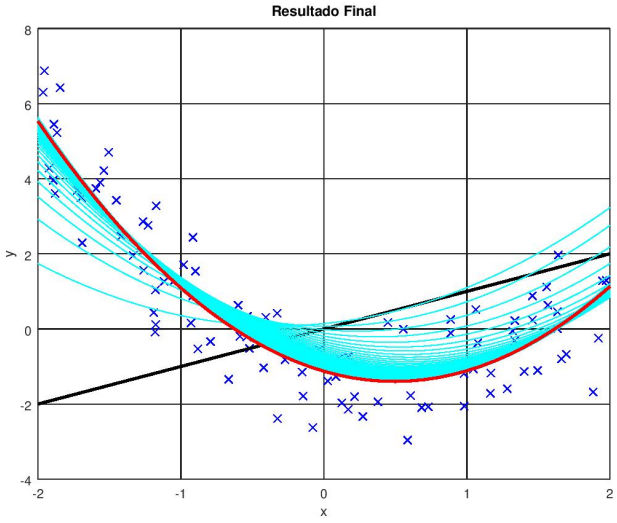

Figura 8. Gráfica del resultado final.

Problema 2:

2.1. Construya la matriz M de (2.1).

Para este punto se siguió lo especificado en el enunciado, aplicando la formula de la matriz M (2.1). Como se ve en la siguiente figura.

```
39  ## ##################
40  ## ## Problema 2.1 ##
41  ## ##################
42  ## Construya la matriz M
43  M  = zeros(dim,4); ## CAMBIE ESTO!!
44  contM=1;
45  while(contM<=dim)
46    M(contM,1) = 1;
47    M(contM,2) = -2 * emisorPos(1,contM);
48    M(contM,3) = -2 * emisorPos(2,contM);
49    M(contM,4) = -2 * emisorPos(3,contM);
50    contM=contM+1;
51  endwhile
```

Figura 2.1. Matriz M

2.2. Construya el vector b de (2.1).

Para hacer este punto se tomó en cuenta las distancias del emisores y la posición de los mismos. Para ello el código quedó como la siguiente figura. Nuevamente se basó en el b del enunciado (2.1).

```
52  ## ##################
53  ## ## Problema 2.2 ##
54  ## ##################
55  ## Construya el vector b
56
57  b = zeros(dim,1);
58  contB=1;
59  while(contB<=dim)
60  b(contB,1) = (dists(1,contB) ^ 2)-((emisorPos(1,contB))^2 + (emisorPos(2,contB))^2 + (emisorPos(3,contB))^2);
61  contB=contB+1;
62  endwhile
```

Figura 2.2. Vector b.

2.3. Utilizando la descomposición de valores singulares (función svd) calcule la matriz seudoinversa de M.

En este punto se utilizó la funcion de octave svd y posteriormente se utilizó la función de octave resize para ajustar los tamaños devueltos y con ello poder calcular la seudoinversa M. Como se aprecia en la siguiente imagen.

```
64  ## ##################
65  ## ## Problema 2.3 ##
66  ## ##################
67
68  ## Calcule la matriz seudo-inversa utilizando SVD
69
70  [U,S,V] = svd(M);
71  Ur = resize(U,dim,4);
72  Vr = resize(V,4,4);
73  Sr = resize(S,4,4);
74  iM =  Vr * inv(Sr) * Ur';
75
76  ## Verifique que iM y pinv(M) son lo mismo
77  if (norm(iM-pinv(M),"fro") > 1e-6)
78    error("Matriz inversa calculada con SVD incorrecta");
79  endif
```

Figura 2.3. Calculo de la matriz seudoinversa de M (iM).

2.4. Calcule la solución particular p̂.

A partir de la ecuación Mp̂ = b, se puede llegar a la solución que es la mostrada en la línea 87 de la siguiente figura.

```
81  ## ##################
82  ## ## Problema 2.4 ##
83  ## ##################
84
85  ## Calcule la solución particular
86  hatp=zeros(4,1); ## CAMBIE ESTO!!
87  hatp = iM * b;
```

Figura 2.4. Solución particular de p̂ (hatp).

2.5. Para el caso en que solo hayan 3 emisores, encuentre las dos posibles soluciones. Note que esto requiere encontrar el vector que engendra el espacio nulo.

Para encontrar el vector que engendra el espacio nulo, se parte del vector Vr, que fue el devuelto por la función svd y que fue aplicado por la función resize.

Posteriormente se obtienen los valores para a, b y c de la ecuación cuadratica que despeja a λ, que es la ecuación 2.2 del enunciado. Además seguidamente se aplica la fórmula general para obtener ya sea la "option 1" o la "option 2" según sea el caso y finalmente que el segundo valor de p̂(hatp2), las soluciones y el vector nulo se obtiene la solución de p.

```
92    ## ##################
93    ## ## Problema 2.5 ##
94    ## ##################
95
96    ## Con 3 emisores, calcule las dos posibles posiciones
97
98    ## PONGA SU CÓDIGO AQUÍ
99
100   hatp2 = pinv(M) * b;
101   vectNulo = [Vr(1,4);Vr(2,4);Vr(3,4);Vr(4,4)];
102   a = vectNulo(2) ^ 2 + vectNulo(3) ^ 2 + vectNulo(4) ^ 2;
103   b = (2 * vectNulo(2) * hatp2(2)+ 2 * vectNulo(3) * hatp2(3) + 2 * vectNulo(4) * hatp2(4) - vectNulo(1));
104   c = ( hatp2(2) ^ 2 + hatp2(3) ^ 2 + hatp2(4) ^ 2 - hatp2(1));
105
106   if(option==1)
107    sol  = (-b + sqrt(b^2 - 4*a*c)) / (2 * a);
108   else
109     sol = (-b - sqrt(b^2 - 4*a*c)) / (2 * a);
110   endif
111   p = [hatp2(1) + sol*vectNulo(1);hatp2(2) + sol*vectNulo(2); hatp2(3) + sol*vectNulo(3); hatp2(4) + sol*vectNulo(4)];
```

Figura 2.5. Obtención de la solución p.

2.6. Para el caso con 4 o más emisores, estime la posición del objeto.

En este caso se cumple la condición del punto 2.4, ya que el tamaño de la matriz será mayor a 4 y por ende no existirán vectores nulos, por lo que la solución p = hatp(p̂).

```
113   else
114     ## ##################
115     ## ## Problema 2.6 ##
116     ## ##################
117
118     ## Caso general
119
120     ## PONGA SU CÓDIGO AQUÍ
121     p = hatp;
122   endif
```

Figura 2.6. Obtención de la solución p para 4 o maś emisores.

2.7. Verifique que las posiciones calculadas para la trayectoria descrita a través de las distancias efectivamente se encuentran a esas distancias.

En este punto se obtienen las posiciones de las soluciones p (mediante el puntero/contador i) y la posición de los emisores (mediante el puntero/contador i). A partir de esos valores se empieza a calcular el valor de la distancia entre ellos, mediante la clásica ecuación para la distancia entre puntos. Como se puede apreciar en la siguiente figura.

```
154   ## ##################
155   ## ## Problema 2.7 ##
156   ## ##################
157
158   ## Calcule la distancia a cada emisor, dada la posición "pos" del
159   ## objeto y las posiciones de los emisores.
160
161   ## PONGA SU CÓDIGO AQUÍ
162
163   for i = 1:rows(pos)
164     for j = 1:5
165       xe = emisorPos(1,j);
166       ye = emisorPos(2,j);
167       ze = emisorPos(3,j);
168       d(i,j) = sqrt((pos(i,2)-xe)^2 + (pos(i,3)-ye)^2 + (pos(i,4)-ze)^2); # distancia
169     endfor
170   endfor
171   d
172 endfunction
```

Figura 2.7. Obtención de la distancia entre las soluciones y los emisores.

Las siguientes imágenes muestran el resultado de las gráficas:

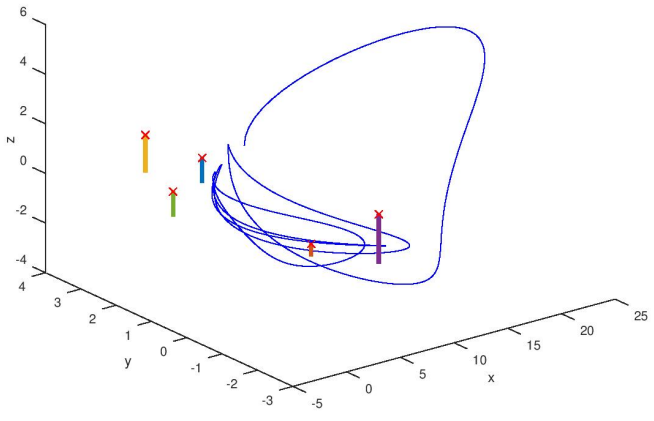

Figura 2.8. Trayectoria estimada y emisores.

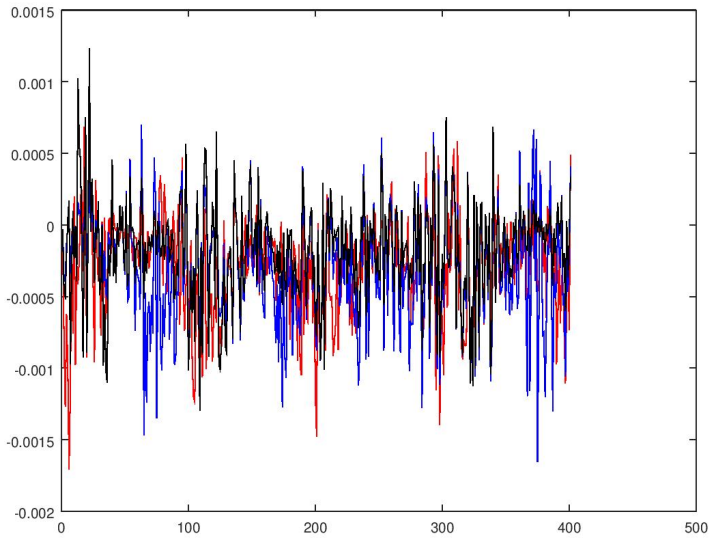

Figura 2.9. Errores.

Problema 3:

3.1. Grafique los puntos tridimensionales almacenados en X con “×” azules.

Para realizar este punto se utiliza la función plot3 y el resultado se muestra en la siguiente figura.

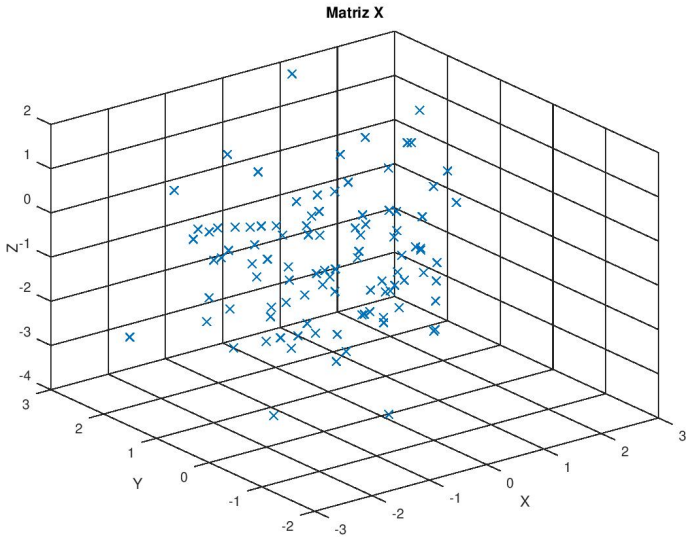

Figura 3.1. Gráfica de los puntos de X.

3.2. Indique cómo se calcula matemáticamente el punto medio de los datos, y utilice GNU/Octave para determinarlo. Recuerde que los datos son tridimensionales y se encuentran en las columnas de X.

Como se explica al principio de la figura 3.2 el valor medio se calcula dividiendo entre el total de datos N, el resultado de la sumatoria desde i=1 hasta N de los datos de X.
Por ello en el código se van sumando cada componente de x,y,z y se multiplica por (1/N);

```
32  ##############################################
33  ## Problema 3.2                              ##
34  ## Calcule la media de los datos             ##
35  ##############################################
36
37  # El valor medio se calcula de la siguiente manera:
38  # u = (1/n)sum_{i=1}^{n}{Xi}
39  function [result] = valorMedio(N, X)
40    totX=0;
41    totY=0;
42    totZ=0;
43    cont=1;
44    while(cont<=N)
45    totX= totX + X(1,cont);
46    totY= totY + X(2,cont);
47    totZ =totZ + X(3,cont);
48    cont+=1;
49    endwhile
50    Ux=(1/N)*(totX);
51    Uy=(1/N)*(totY);
52    Uz=(1/N)*(totZ);
53    result = [Ux,Uy,Uz];
54  endfunction
```

Figura 3.2. Código del punto 3.2

3.3. Muestre en la misma gráfica de 3.1 el punto medio con un círculo rojo.

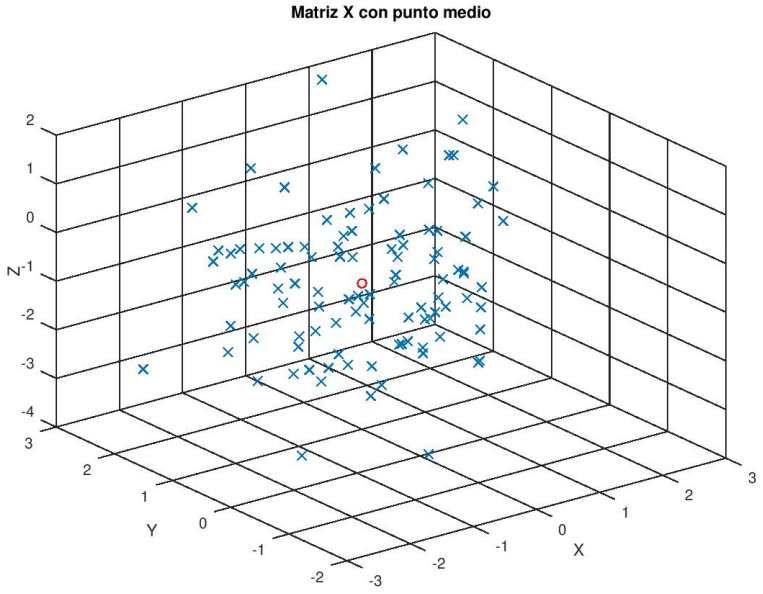

Figura 3.3. Gráfica del punto 3.1 con el valor medio (circulo medio).

3.4. Determine una nueva matriz $\bar{X}$ con datos de media cero. Grafique en otra figura dichos datos con "×" azules.

Para realizar este punto primero se llama a la función del punto anterior (punto medio U) y a cada valor de X se le resta el valor de U, ya sea en x,y,z. Posteriormente se gráfica y el resultado se muestra en la imagen 3.5

```
75  ##############################################
76  ## Problema 3.4                             ##
77  ## Calcule los datos sin media              ##
78  ##############################################
79  [U]=valorMedio(columns(X), X);
80  xMed=[];
81  yMed=[];
82  zMed=[];
83  cont=1;
84  while(cont<=N)
85  xMed=[xMed;(X(1,cont)-U(1))];
86  yMed=[yMed;(X(2,cont)-U(2))];
87  zMed=[zMed;(X(3,cont)-U(3))];
88  cont+=1;
89  endwhile
90  XM0=[xMed, yMed, zMed]';
91  figure(3);
92  plot3(XM0(1,1:end),XM0(2,1:end),XM0(3,1:end),'x');
93  hold on;
94  grid on;
95  title('Matriz X de Media Cero');
96  xlabel("X")
97  ylabel("Y")
98  zlabel("Z")
99  hold off;
```

Figura 3.4. Código para calcular los datos de media cero.

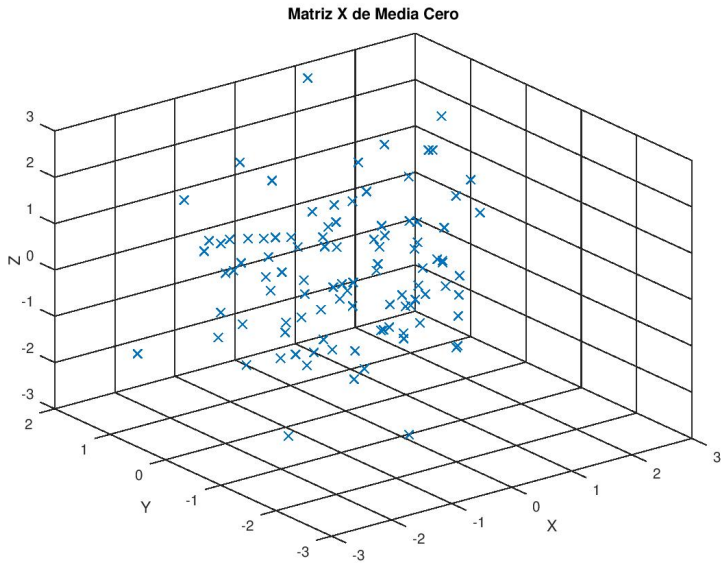

Figura 3.5. Resultado de la gráfica de los datos de media cero.

3.5. Indique cómo se calcula matemáticamente la matriz de covarianza $\Sigma_{\bar{X}}$ de los datos $\bar{X}$, y determine dicha matriz utilizando GNU/Octave.

Tal y como se indica en la primera parte de la siguiente imagen la matriz de covarianza se obtiene multiplicando la matriz por su transpuesta y luego por (1/N).

```
102 ################################################
103 ## Problema 3.5                                ##
104 ## Calcule la matriz de covarianza             ##
105 ################################################
106 # La matriz de covarianza se calcula de la siguiente manera:
107 # MatCov = (1/n)*X*X^{T}
108
109 XMatCov=(XM0*XM0').*(1/N)
110
```

Figura 3.6. Código para calcular la matriz de covarianza.

3.6. Encuentre los eigenvalores y eigenvectores de la matriz de covarianza utilizando GNU/Octave.

Debido a que se dice en la pregunta que se calculen los eigenvalores y eigenvectores mediante GNU/Octave se utiliza la función provista por Octave para dicho caso (eig) y se guardan en las variables eigenvectores y eigenvalores según corresponda.

```
111 ################################################
112 ## Problema 3.6                                ##
113 ## Encuentre los eigenvalores y eigenvectores ##
114 ################################################
115
116 [eigenvectores, eigenvalores]=eig(XMatCov)
```

Figura 3.7. Código para calcular los eigenvalores y eigenvectores.

3.7. Reordene la matriz de eigenvalores y eigenvectores, de manera que sea conveniente para realizar el análisis de componentes principales.

Los eigenvalores no se ordenan ya que como son una matriz diagonal ya están ordenados, para ordenar los eigenvectores se utiliza la transpuesta de ese dato.

```
118 ################################################
119 ## Problema 3.7                                ##
120 ## Reordene los eigenvectores para PCA         ##
121 ################################################
122
123 eigVecComponentesPrincipales=eigenvectores'
```

Figura 3.8. Código para reordenar los eigenvectores.

3.8. Indique explícitamente cuáles son los ejes principales y qué varianza tienen los datos en esos ejes.

Para obtener los ejes principales se debe partir de los eigenvectores ordenados y cada eje corresponde a la fila respectiva de los componentes principales, por ejemplo el eje principal 1 es la fila 1 y así sucesivamente.

Ahora, para obtener la varianza de cada eje se le asigna el valor de los eigenvalores según sea el orden, por ejemplo para la varianza del eje 1 se toma el eigenvalor de la posición (1,1) y así sucesivamente.

```
125 ##########################################################################
126 ## Problema 3.8                                                         ##
127 ## Cuáles son los ejes principales y qué varianza tiene los datos       ##
128 ##########################################################################
129
130 EjePrincipal1=eigVecComponentesPrincipales(1, 1:end)
131 EjePrincipal2=eigVecComponentesPrincipales(2, 1:end)
132 EjePrincipal3=eigVecComponentesPrincipales(3, 1:end)
133
134 varianzaEje1=eigenvalores(1,1)
135 varianzaEje2=eigenvalores(2,2)
136 varianzaEje3=eigenvalores(3,3)
```

Figura 3.9. Código para obtener los ejes principales y la varianza de cada eje.

3.9. Calcule la proyección de los datos en un plano engendrado por los dos primeros ejes principales. Grafique los puntos proyectados sobre el plano en una figura aparte.

La proyección Y se calcula multiplicando los ejes principales por la matriz de media cero. Y el resultado de esa operación se muestra en la imagen 3.11.

```
138 ##########################################################################
139 ## Problema 3.9                                                          ##
140 ## Calcule la proyección de los datos al plano engendrado por los dos ##
141 ## eigenvectores                                                         ##
142 ##########################################################################
143
144 #Proyección: Y=PX
145 Y=[EjePrincipal1;EjePrincipal2]*XM0;
146 ## Grafique la proyección
147 figure(4);
148 plot(Y(1,1:end),Y(2,1:end), '.');
149 hold on;
150 grid on;
151 title('Puntos proyectados sobre el plano');
152 xlabel("X")
153 ylabel("Y")
154 hold off;
```

Figura 3.10. Código para obtener la proyección del plano engendrado por los eigenvectores.

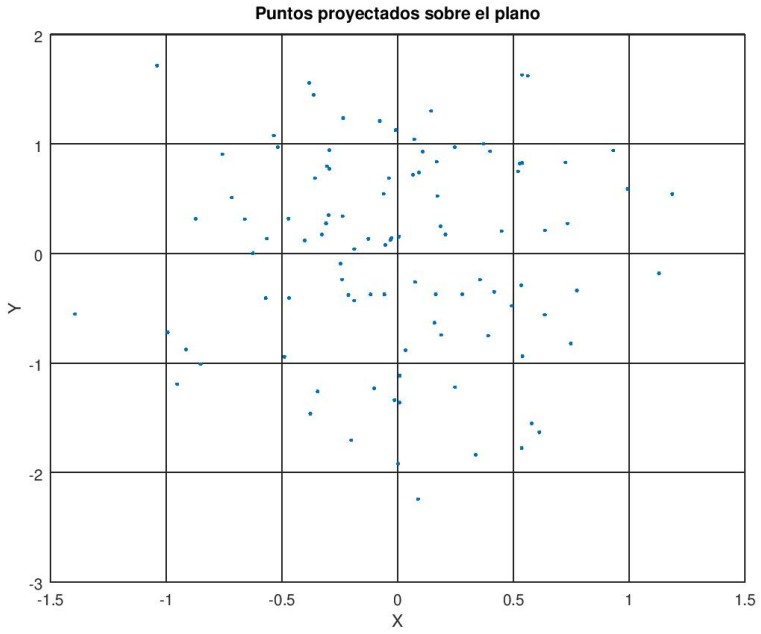

Figura 3.11. Gráfica del plano engendrado por los dos primeros ejes principales.

3.10. Reconstruya a partir de los puntos bidimensionales obtenidos con la proyección anterior, los puntos tridimensionales correspondientes en el espacio original, donde están los datos de entrada originales cargados del archivo. Grafique las reconstrucciones en una nueva figura, junto a los datos originales de entrada. Use "×" azules para los datos originales y cuadrados magenta para los datos reconstruidos.

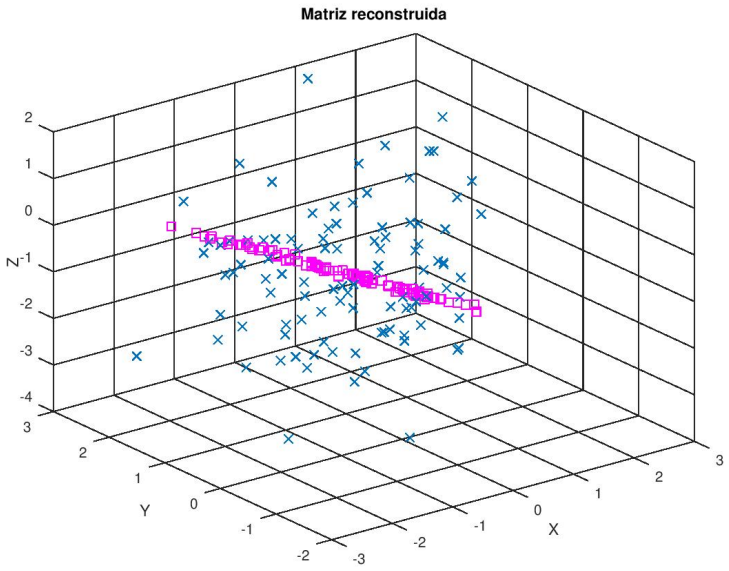

Figura 3.12. Reconstrucción de los puntos originales a partir de la proyección.