ENTREGA 4 y 5

Apellidos, Nombre:	Abengózar Vilar, Diego José
Apellidos, Nombre:	García Castellanos, Alejandro

Observación: no hemos añadido el error relativo por posibles fallos de división por cero al trabajar con matrices posiblemente incompletas.

Método 1: Potencia
Método 2: Mínimos cuadrados logarítmico
Método 3: Mínimos cuadrados ponderado
Método 4: Mínimo suma de desviaciones logarítmico
Método 5: Mínimo suma de desviaciones ponderado

CASO 1: un caso consistente

Código:

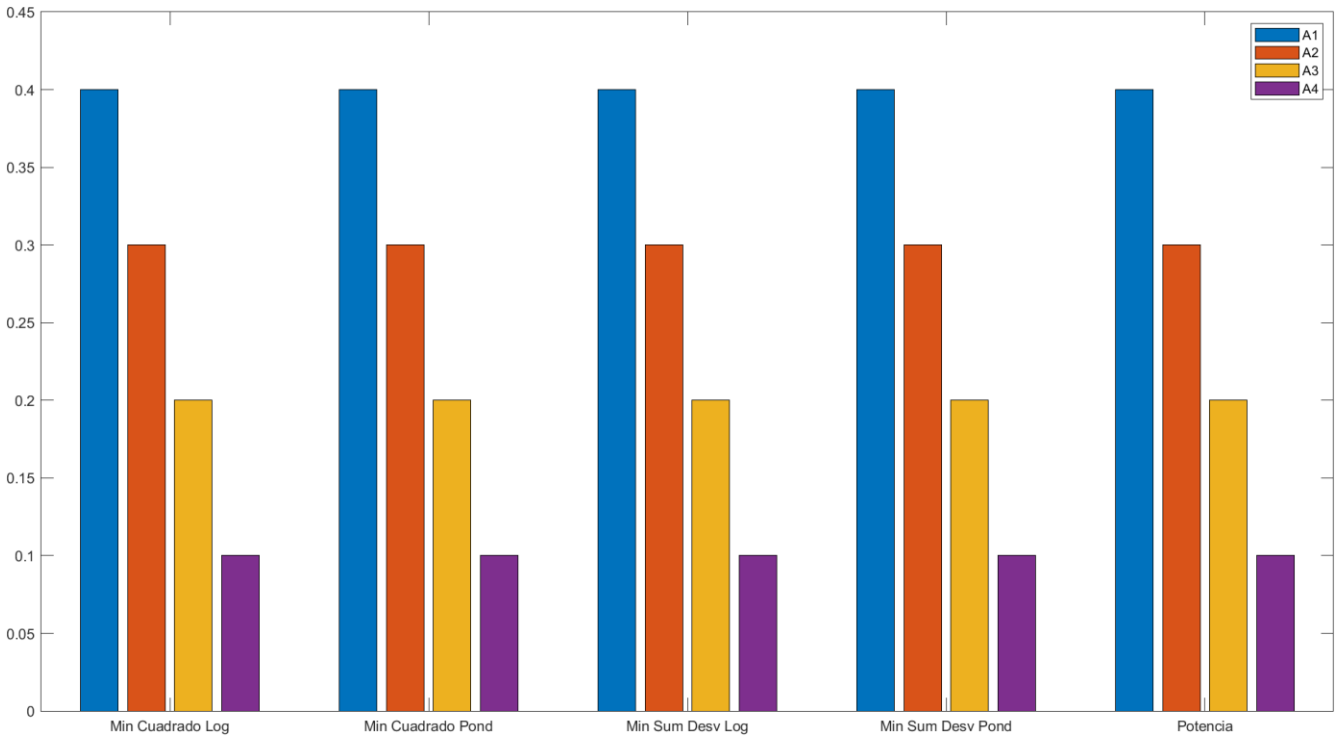
```
clc;
clear;

fprintf("-------------Consistente-------------\n");
w = [0.4 0.3 0.2 0.1]
M= zeros(4);
for i = 1:4
    for j = 1:4
        M(i,j) = w(i)/w(j);
    end
end

M
ic(M)

fprintf("-------------Metodo de la potencia-------------\n");
w0 = funciones(0, M) % potencia(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w0, M)

fprintf("-------------Metodo de Minimos Cuadrados [Log]-------------\n");
w1 = funciones(1, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w1, M)

fprintf("-------------Metodo de Minimos Cuadrados [Ponderado]-------------\n");
w2 = funciones(2, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w2, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Log]-------------\n");
w3 = funciones(3, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w3, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Ponderado]-------------\n");
w4 = funciones(4, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w4, M)
```

```
%Dibujar pesos
figure();
c = categorical(["Potencia", "Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv
Pond"]);
bar(c, [w0';w1';w2';w3';w4']);
l=compose("A%d",(1:length(M)));
legend(l);
```

Tabla de resultados y medidas de error

IC(M) = 0	w	Ranking	Max Residuo	Error Fr	Error Norm1
MÉTODO 1	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	8.8818e-16	1.0489e-15	1.9429e-15
MÉTODO 2	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	8.8818e-16	1.0551e-15	2.0817e-15
MÉTODO 3	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	1.3323e-15	2.6210e-15	7.1887e-15
MÉTODO 4	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	4.4409e-16	5.6882e-16	1.1657e-15
MÉTODO 5	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	8.8818e-16	1.1501e-15	2.6368e-15

Como la matriz es consistente podemos observar que al ser el caso ideal los cinco métodos nos dan el mismo resultado y contienen errores bastantes parecidos y prácticamente despreciables, ya que se aproximan a la precisión máxima de la máquina. También se puede observar que estos errores son tan bajos que dependiendo de la ejecución pueden variar, dando situaciones donde se considere que el error es 0,0.



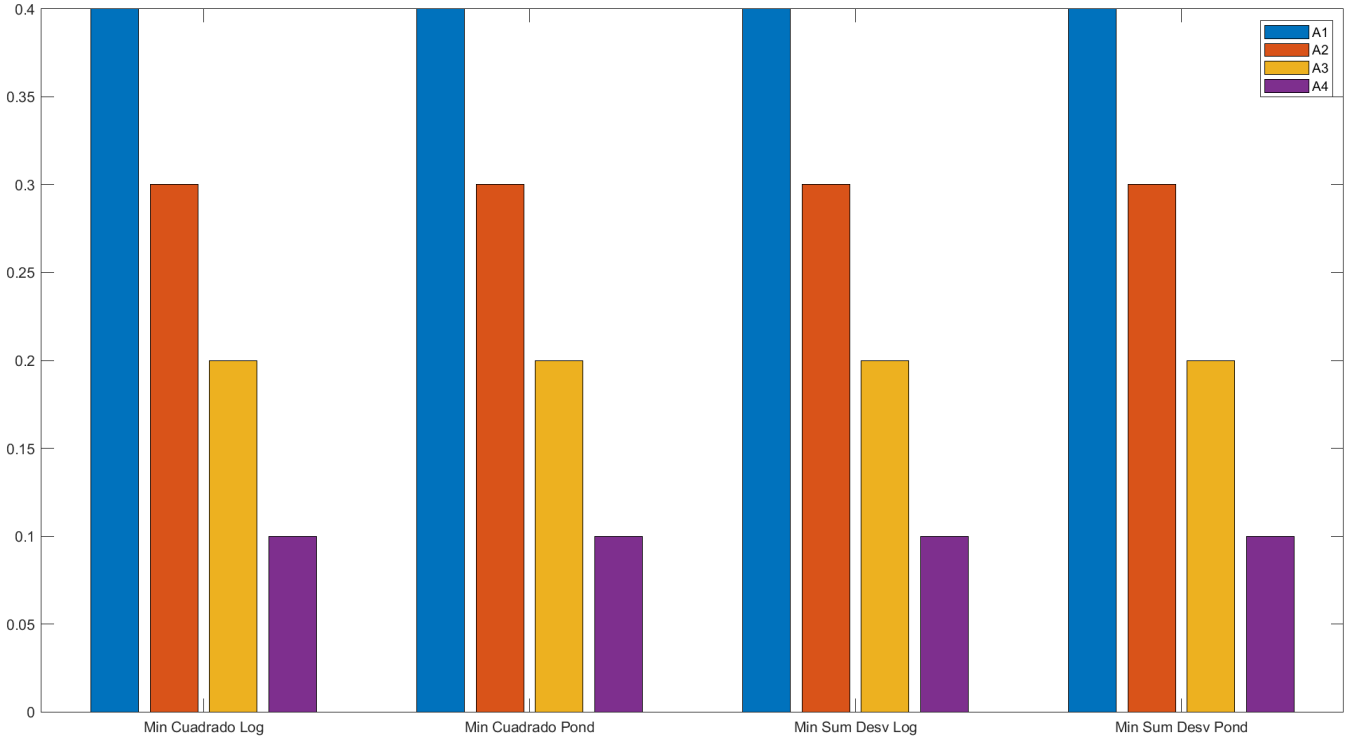
CASO 2. Un caso consistente con varios expertos

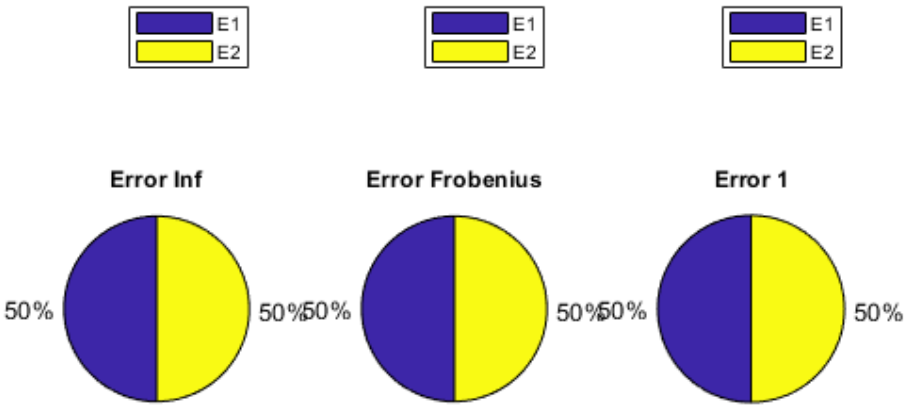
```
clear;
fprintf("-------------Consistente Varios Expertos-------------\n");
w = [0.4 0.3 0.2 0.1]
M= zeros(4);
for i = 1:4
    for j = 1:4
        M(i,j) = w(i)/w(j);
    end
end

M
ic(M)

fprintf("-------------Metodo de Minimos Cuadrados [Log]-------------\n");
w1 = funciones(1, M, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w1, M, M)

fprintf("-------------Metodo de Minimos Cuadrados [Ponderado]-------------\n");
w2 = funciones(2, M, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w2, M, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Log]-------------\n");
w3 = funciones(3, M, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w3, M, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Ponderado]-------------\n");
w4 = funciones(4, M, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w4, M, M)

%Dibujar pesos
figure();
c = categorical(["Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv Pond"]);
bar(c, [w1';w2';w3';w4']);
l=compose("A%d",(1:length(M)));
legend(l);
```

Tabla de resultados y medidas de error

	w	Ranking	Max Residuo	Error Fr	Error Norm1
MÉTODO 2	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	8.327e-16	1.0551e-15	2.0817e-15
MÉTODO 3	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	1.3323e-15	2.6210e-15	7.1887e-15
MÉTODO 4	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	4.4409e-16	5.6882e-16	1.1657e-15
MÉTODO 5	(0.4000,0.3000,0.2000,0.1000)	(A1, A2, A3, A4)	8.8818e-16	1.1501e-15	2.6368e-15

En este ejemplo hemos considerado dos expertos con las mismas decisiones. Claramente obtenemos los mismos resultados que en caso anterior.





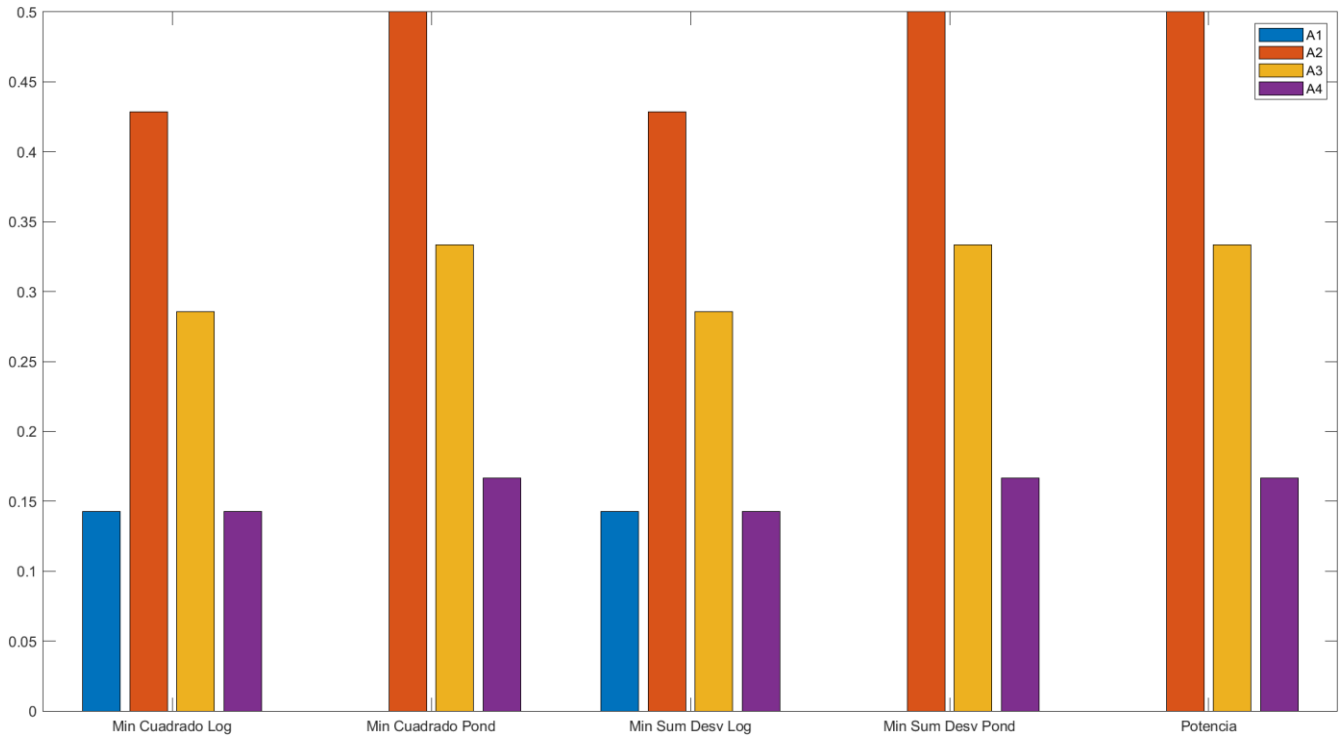
Para todos los métodos obtenemos los mismos resultados que en la figura anterior en la que se comparan los errores de los distintos expertos. Esto se debe a que los expertos tienen exactamente la misma “opinión”.

CASO 3. Un caso consistente incompleto

```
clear;

fprintf("-------------Consistente Incompleta-------------\n");
w = [0.4 0.3 0.2 0.1]
M= zeros(4);
for i = 1:4
    for j = 1:4
        M(i,j) = w(i)/w(j);
    end
end

%El experto no opina sobre la opcion 1
M(1,:) = 0;
M(:,1) = 0;
% M =

         0         0         0         0
         0    1.0000    1.5000    3.0000
         0    0.6667    1.0000    2.0000
         0    0.3333    0.5000    1.0000

M
ic(M)

fprintf("-------------Metodo de la potencia-------------\n");
w0 = funciones(0, M) % potencia(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w0, M)

fprintf("-------------Metodo de Minimos Cuadrados [Log]-------------\n");
w1 = funciones(1, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w1, M)

fprintf("-------------Metodo de Minimos Cuadrados [Ponderado]-------------\n");
w2 = funciones(2, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w2, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Log]-------------\n");
w3 = funciones(3, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w3, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Ponderado]-------------\n");
w4 = funciones(4, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w4, M)

%Dibujar pesos
figure();
c = categorical(["Potencia", "Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv
Pond"]);
bar(c, [w0';w1';w2';w3';w4']);
l=compose("A%d",(1:length(M)));
legend(l);
```

Tabla de resultados y medidas de error

	w	Ranking	Max Residuo	Error Fr	Error Norm1
MÉTODO 1	(0,0.5,0.3333,0.1667)	(A2, A3, A4, A1)	Inf	NaN	NaN
MÉTODO 2	(0.1429,0.4286,0.2857,0.1429)	(A2, A3, A4 = A1)	0.3333	0.4494	0.9815
MÉTODO 3	(0,0.5,0.3333,0.1667)	(A2, A3, A4, A1)	Inf	NaN	NaN
MÉTODO 4	(0.1429,0.4286,0.2857,0.1429)	(A2, A3, A4 = A1)	0.3333	0.4494	0.9815
MÉTODO 5	(0,0.5,0.3333,0.1667)	(A2, A3, A4, A1)	Inf	NaN	NaN



Como hemos quitado la valoración de la opción 1, vemos que el resultado ha desplazado el ranking una posición. Cabe destacar que el ranking de los métodos logarítmicos antes de hacer la transformación exponencial sería v = [0, 1.0986, 0.6931, 0], lo que da un orden (A2, A3, A4, A1) que coincide con el resto de los métodos, como debería ocurrir a partir de una matriz consistente.
Los errores obtenidos no son fieles a la realidad porque no consideran que la matriz original es incompleta.

CASO 4. Un caso no consistente

Código:

```
clear;


fprintf("-------------NO Consistente-------------\n");
M = [1.0000   0.1429    0.1429    0.2000;
    7.0000    1.0000    0.5000    0.3333;
    7.0000    2.0000    1.0000    0.1111;
    5.0000    3.0000    9.0000    1.0000]

%M2 = [1 2,2,4;1,1,1.5,3;0.5,0.6,1,2;0.25,0.3,0.5,1]

ic(M)

fprintf("-------------Metodo de la potencia-------------\n");
w0 = funciones(0, M) % potencia(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w0, M)

fprintf("-------------Metodo de Minimos Cuadrados [Log]-------------\n");
w1 = funciones(1, M) % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w1, M)
fprintf("-------------Metodo de Minimos Cuadrados [Ponderado]-------------\n");
w2 = funciones(2, M) % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w2, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Log]-------------\n");
w3 = funciones(3, M) % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w3, M)

fprintf("-------------Metodo de Minimo Suma Desviaciones [Ponderado]-------------\n");
w4 = funciones(4, M) % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w4, M)

%Dibujar pesos
figure();
c = categorical(["Potencia", "Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv
Pond"]);
bar(c, [w0';w1';w2';w3';w4']);
l=compose("A%d",(1:length(M)));
legend(l);
```

Tabla de resultados y medidas de error

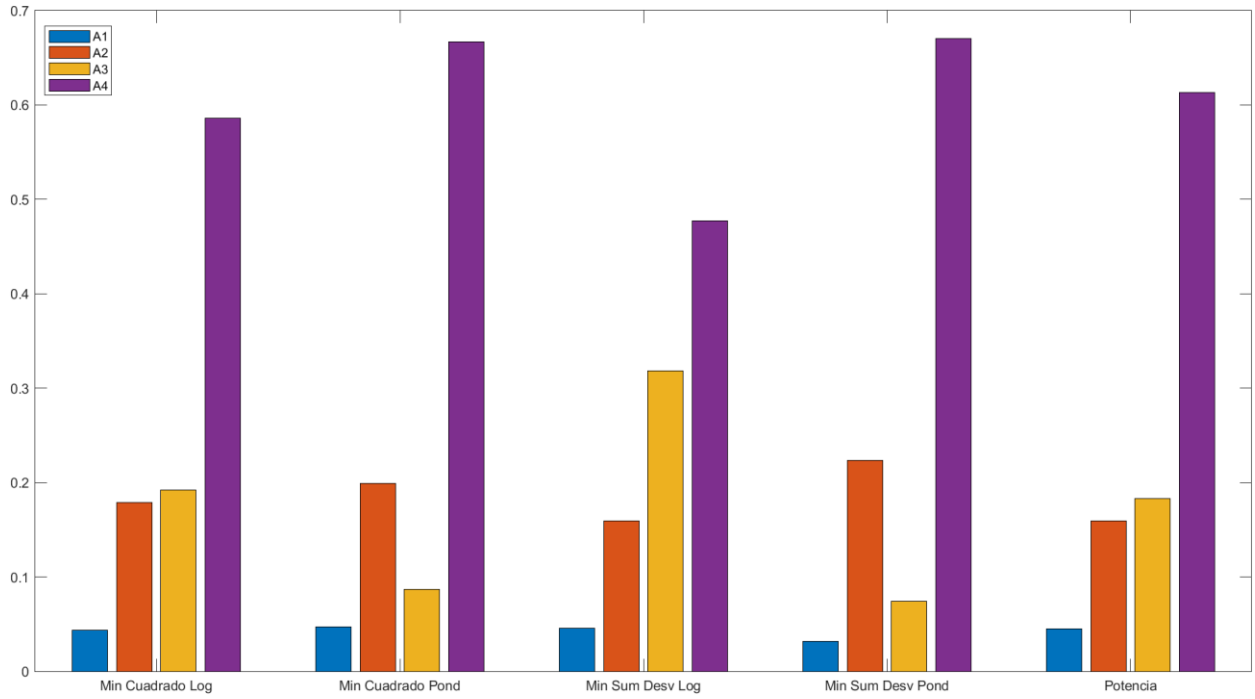
IC(M)= 0.2761	w	Ranking	Max Residuo	Error Fr	Error Norm1
MÉTODO 1	(0.0448,0.1592,0.1829,0.6132)	(A4, A3, A2, A1)	0.5429	0.7110	1.4625
MÉTODO 2	(0.0435,0.1786,0.1920,0.5859)	(A4, A3, A2, A1)	0.5303	0.6948	1.3809
MÉTODO 3	(0.0470,0.1990,0.0869,0.6671)	(A4, A2, A3, A1)	0.5746	0.7027	1.4259
MÉTODO 4	(0.0455,0.1591,0.3182,0.4773)	(A4, A3, A2, A1)	0.4688	0.6221	1.0815
MÉTODO 5	(0.0319,0.2234,0.0745,0.6702)	(A4, A2, A3, A1)	1	1.0587	1.5795

Los matrices inconsistentes las hemos obtenido del siguiente trabajo http://www.biblioteca.uma.es/bbldoc/articulos/16641152.pdf, donde se obtiene un método para obtener matrices inconsistentes y que mantengan la reciprocidad.

Como podemos observar al no tener un índice de consistencia igual a cero sabemos que la matriz no es consistente, y esto da lugar a los resultados obtenidos.

Al linealizar el sistema para poder aplicar el método de mínimos cuadrados nos damos con la situación de que no estamos minimizando realmente el problema original, sino el original una vez aplicado la transformación correspondiente. Es por esto por lo que los dos métodos de mínimos cuadrados difieren las soluciones uno de otro; y sobre todo podemos observar que es mayor el del logaritmo.

Es claro que en los métodos que minimizan cierta norma tienen un menor error en su correspondiente error lo cual nos indica que funcionan correctamente. No es claro el motivo por el que el método 5 (suma desviaciones ponderado) tiene un error mayor que los otros métodos.



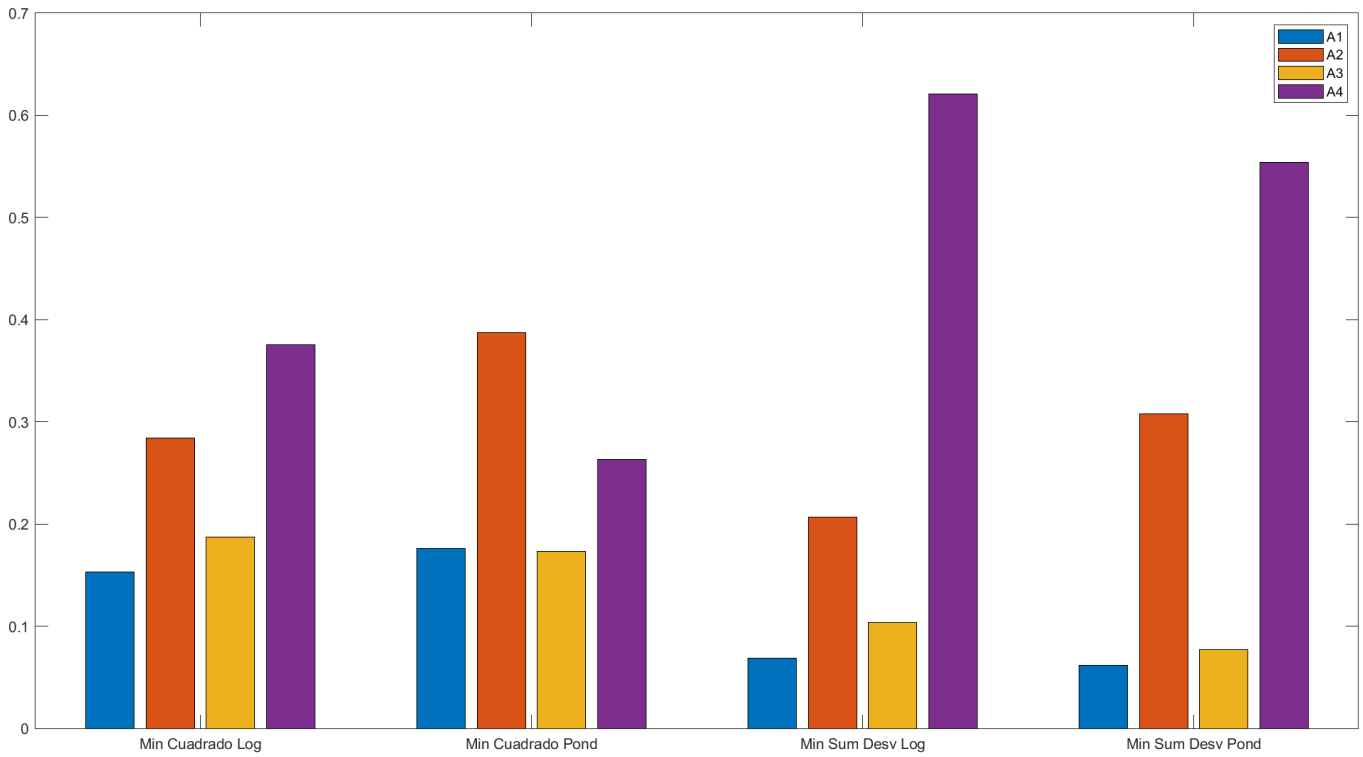
CASO 5: Un caso no consistente con varios expertos

```
clear;
fprintf("-------------NO Consistente Varios Expertos-------------\n");
M1 = [1.0000    0.1429    0.1429    0.2000;
    7.0000    1.0000    0.5000    0.3333;
    7.0000    2.0000    1.0000    0.1111;
    5.0000    3.0000    9.0000    1.0000]

M2 = [1 1/5 1/3 1/9;
      5  1   4  1/8;
      3  1/4 1  1/9;
      9   8  9   1]

M3 = [1 1/3 1/7 1/9;
      3  1  1/2 1/5;
      7  2   1  1/7;
      9  5   7   1]

%Tomamos M3 traspuesta
ic(M1)
ic(M2)
ic(M3')

fprintf("-------------Metodo de Minimos Cuadrados [Log]-------------\n");
w1 = funciones(1, M1, M2, M3') % minCuadLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w1, M1, M2, M3')

fprintf("-------------Metodo de Minimos Cuadrados [Ponderado]-------------\n");
w2 = funciones(2, M1, M2, M3') % minCuadPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w2, M1, M2, M3')

fprintf("-------------Metodo de Minimo Suma Desviaciones [Log]-------------\n");
w3 = funciones(3, M1, M2, M3') % minSumDesvLog(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w3, M1, M2, M3')

fprintf("-------------Metodo de Minimo Suma Desviaciones [Ponderado]-------------\n");
w4 = funciones(4, M1, M2, M3') % minSumDesvPond(M)
[errorInf, IndexMaxErr, errorFro, errorUno,  errorNoAciertos] = errores(w4, M1, M2, M3')

%Dibujar pesos
figure();
c = categorical(["Min Cuadrado Log", "Min Cuadrado Pond", "Min Sum Desv Log", "Min Sum Desv Pond"]);
bar(c, [w1';w2';w3';w4']);
l=compose("A%d",(1:length(M)));
legend(l);
```

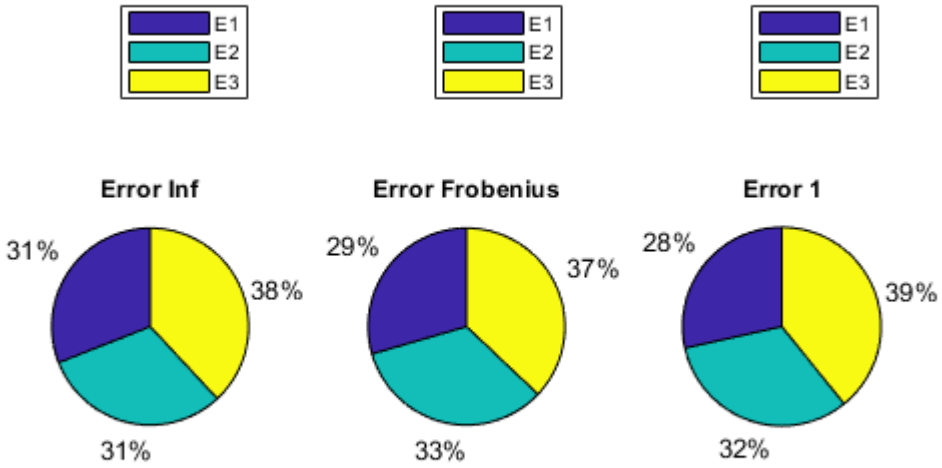
Tabla de resultados y medidas de error

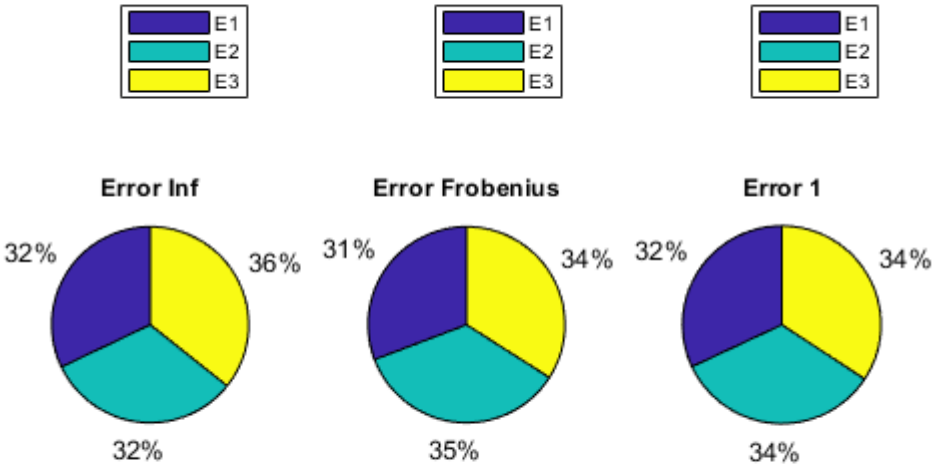
	w	Ranking	Max Residuo	Error Fr	Error Norm1
MÉTODO 2	(0.1529,0.2842,0.1872,0.3758)	(A4, A2, A3, A1)	0.5371	1.3561	5.8309
MÉTODO 3	(0.1761,0.3876,0.1731,0.2632)	(A2, A4, A1, A3)	0.5207	1.3940	6.0123
MÉTODO 4	(0.0690,0.2069,0.1034,0.6207)	(A4, A2, A3, A1)	0.5556	1.3190	5.3624
MÉTODO 5	(0.0615,0.3077,0.0769,0.5538)	(A4, A2, A3, A1)	0.5556	1.3539	5.4058

Vemos que los métodos que minimizan norma 2 hacen más “homogéneos” los resultados que los métodos que minimizan norma 1, esto lo podemos observar al ver que 2 de los expertos tienen preferencia por la opción 4 mientras que el tercer experto tiene preferencia por la opción 1 y los métodos de mínimos cuadrados representan mejor la visión general del grupo, mientras que los métodos que usan el simplex dan mucha más importancia a la opción que más expertos quieren sin tener tanto en cuenta el resto.

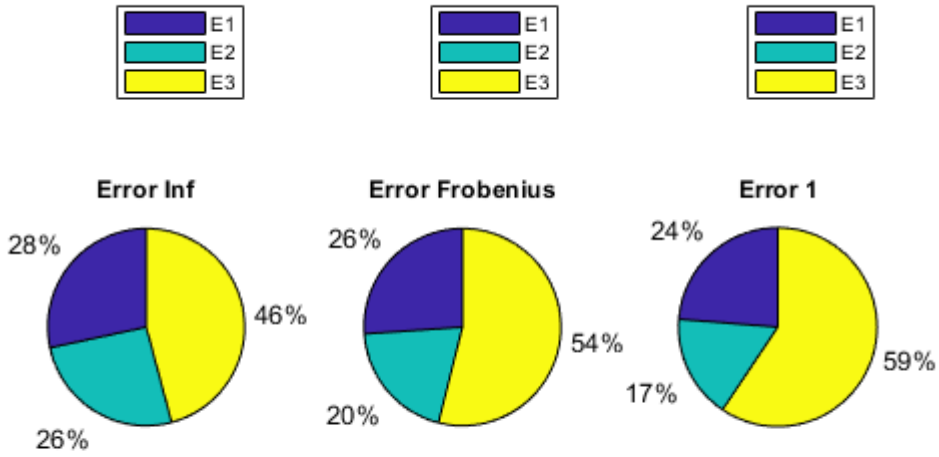


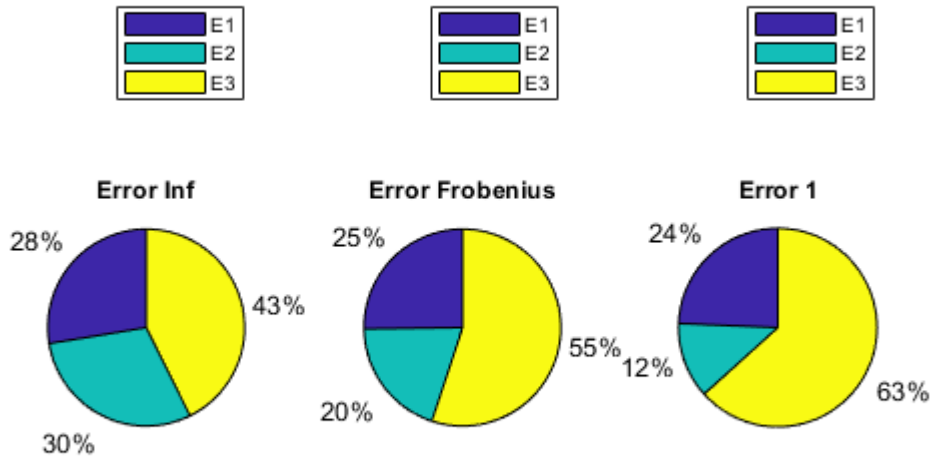
Métodos 2 y 3


E1
E2
E3
Error Inf
31%
38%
31%
E1
E2
E3
Error Frobenius
29%
37%
33%
E1
E2
E3
Error 1
28%
39%
32%


E1
E2
E3
Error Inf
32%
36%
32%
E1
E2
E3
Error Frobenius
31%
34%
35%
E1
E2
E3
Error 1
32%
34%
34%

Métodos 4 y 5





Vemos en las gráficas de los expertos que, en los métodos de mínimos cuadrados, los expertos tienen errores parecidos ya que los métodos los intentan equilibrar, mientras que en los métodos del símplex los errores de los expertos son más dispares porque elige el experto que más minimiza el error y lo prioriza y así sucesivamente.

ANEXO:

funciones.m

```
function w = funciones(metodo, varargin)

 % Juntar las matrices de los expertos
    M = varargin{1};
    for i = 2 : nargin-1
        M = [M; varargin{i}];
    end

if nargin == 1
    [~, w, ~] = potencia(M);
else
    if metodo == 0 && nargin == 2
        [~, w, ~] = potencia(M);
    elseif metodo == 0 && nargin >= 2
        fprintf("Número erróneo de argumentos\n");
    elseif metodo == 1
        w = minCuadLog(varargin, nargin-1);
    elseif metodo == 2
        w = minCuadPond(varargin, nargin-1);
    elseif metodo == 3
        [w, ~, ~] = minSumDesvLog(varargin, nargin-1);
    else
        [w, ~, ~] = minSumDesvPond(varargin, nargin-1);
    end
end

return
```

potencia.m

```
function [lambda,x,iter] = potencia(A,tol,nmax,x0)
% Calcula el mayor (abs) autovalor lambda de A y un autovector asociado x

n=size(A,1);

if nargin == 1
    tol = 1e-06; % Tolerancia
    x0=rand(n,1); % Vector de arranque
    nmax=100; % NÂº mÃ¡x iteraciones
end

x0=x0/norm(x0); x1=A*x0;
lambda=x0'*x1;
err = tol*abs(lambda) + 1;
iter=0;

while err > tol*abs(lambda) && abs(lambda) ~= 0 && iter <= nmax
    x=x1; x=x/norm(x);
    x1=A*x; lambda_new=x'*x1;
    err = abs(lambda_new - lambda);
    lambda=lambda_new;
```

```
    iter = iter + 1;
end

x = x / sum(x);

end
```

minCuadLog.m

```
function w = minCuadLog(E, n)

H = [];
b = [];


for i = 1:n
    M = E{i};

    s=size(M);
    long = (s(1)*s(2))-s(1);

    HAux = zeros([long,s(2)]);
    bAux = zeros([long,1]);

    k = 1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                notZero = (log(M(i,j)) ~= -Inf);
                % Generar H
                HAux(k,i) = notZero;
                HAux(k,j) = -1*notZero ;
                % Generar b
                if (notZero)
                    bAux(k) = log(M(i,j));
                else
                    bAux(k) = 0;
                end
                k = k+1;
            end
        end
    end
    H = [H; HAux];
    b = [b; bAux];
end

% Resolver con minimos cuadrados
v=H\b;
% Deshacer el cambio de logaritmo.
w=exp(v)
% Normalizar
w=w/sum(w);
return
```

minCuadPond.m

```
function w = minCuadPond(E, n)

H = [];
b = [];

for i = 1:n
    M = E{i};

    s = size(M);
    long = (s(1)*s(2))-s(1);

    HAux = zeros([long,s(2)]);

    % Generar b
    bAux=zeros([long,1]);
    %bAux(long+1) = 1;
    b = [b; bAux];

    k = 1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                % Generar
                %(M(i,j) == 0) devuelve 1 o 0 si queremos descartar
                %información que el experto no ha dado
                HAux(k,i) = (M(i,j) ~= 0);
                HAux(k,j) = -M(i,j);
                k=k+1;
            end
        end
    end

    H = [H; HAux];
end

H = [H;  ones(1, s(1))];
b = [b; 1];

% Resolver con minimos cuadrados
w=H\b;

% Normalizar
w=w/sum(w);

return
```

minSumDesvLog.m

```
function [w, n, p] = minSumDesvLog(E, n)

Aeq = [];
beq = [];

s = zeros(1, 2);
```

```
for i = 1:n
    M = E{i};

    s = size(M);
    longM = (s(1)*s(2))-s(1);

    % No hay desigualdades
    A = [];
    b = [];

    AeqAux = zeros(longM, s(2));
    beqAux = zeros(longM, 1);

    k=1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                notZero = (log(M(i,j)) ~= -Inf);
                % Generar Aeq
                AeqAux(k,i) = notZero;
                AeqAux(k,j) = -1*notZero ;
                % Generar beq
                if (notZero)
                    beqAux(k) = log(M(i,j));
                else
                    beqAux(k) = 0;
                end
                k = k + 1;
            end
        end
    end


    Aeq = [Aeq; AeqAux];
    beq = [beq; beqAux];
end

long = size(Aeq, 1);

%Añadimos las metas a las ecuaciones
N = eye(long);
P = -1 * eye(long);

Aeq = [Aeq N P];

x = zeros(s(1) + long + long, 1);
f = [zeros(s(1),1); ones(long,1); ones(long,1)];

% Todos los elementos de x mayor o igual que 0
lb = zeros(length(x), 1);

ub = [];

x = linprog(f, A, b, Aeq, beq, lb, ub);

% Coger de x la parte que nos interesa
v = x(1:s(1));
% Deshacer el cambio logaritmico
```

```
w = exp(v);
% Normalizar
w = w/sum(w);

n = x(s(1)+1:s(1)+long);
p = x(s(1)+long+1:s(1)+2*long);
end
```

minSumDesvPond.m

```
function [w, n, p] = minSumDesvPond(E, n)

Aeq = [];
beq = [];

s = zeros(1, 2);

for i = 1:n
    M = E{i};

    s = size(M);
    longM = (s(1)*s(2))-s(1);

    % No hay desigualdades
    A = [];
    b = [];

    AeqAux = zeros(longM, s(2));

    k = 1;
    for i = 1:s(1)
        for j = 1:s(2)
            if (i ~= j)
                % Generar Aeq
                AeqAux(k,i) = (M(i,j) ~= 0);
                AeqAux(k,j) = -M(i,j);
                k = k + 1;
            end
        end
    end

    Aeq = [Aeq; AeqAux];
end

long = size(Aeq, 1);

%Añadimos las metas a las ecuaciones
N = eye(long);
P = -1 * eye(long);
Aeq = [[Aeq N P]; [ones(1, s(1)) zeros(1, 2*long)]];

x = zeros(s(1) + long + long, 1);
f = [zeros(s(1),1); ones(long,1); ones(long,1)];

beq = zeros([long+1,1]);
beq(long+1) = 1;
```

```
% Todos los elementos de x mayor o igual que 0
lb = zeros(length(x), 1);

ub = [];

x = linprog(f, A, b, Aeq, beq, lb, ub);

% Coger de x la parte que nos interesa
w = x(1:s(1));

% Normalizar
w = w/sum(w);

n = x(s(1)+1:s(1)+long);
p = x(s(1)+long+1:s(1)+2*long);
end
```

errores.m

```
function [errorInf, IndexMaxErr, errorFro, errorUno, errorNoAciertos] = errores(w, varargin)

datos_conocidos_total = 0;
errorInf = [];
IndexMaxErr = [];
errorFro = [];
errorUno = [];
errorNoAciertos = [];

M = [];
W =[];

% Construccion W
WAux = zeros(size(varargin{1}));
for i = 1:size(w)
    for j = 1:size(w)
        WAux(i,j) = w(i)/w(j);
    end
end

% Errores de cada experto
for k = 1:nargin


    if k == nargin
        E = M;
        WAux = W;
    else
        E = varargin{k};

        % Contar 0s de M
        datos_conocidos = 0;
        for m = 1:length(E)
            for n = 1:length(E)
                if E(m,n) ~= 0
```

```
                    datos_conocidos = datos_conocidos + 1;
                end
            end
        end
    end

    % Matriz de residuos
    R = abs(E-WAux) ;

    % Norma Infinito: residuo maximo y su indice
    [errorInfAux, I] = max(R(:));
    errorInf = [errorInf errorInfAux/datos_conocidos];
    [I_row, I_col] = ind2sub(size(R),I);
    IndexMaxErr = [IndexMaxErr [I_row, I_col]];

    % Norma Frobenius
    errorFroAux = norm(R,'fro');
    errorFro = [errorFro errorFroAux/datos_conocidos];

    % Norma 1
    errorUnoAux = sum(sum(R));
    errorUno = [errorUno errorUnoAux/datos_conocidos];

    % Norma "ErrorRel"
    % errorErrRelAux = norm(R./M, 'fro'); Puede dividir por 0
    % errorErrRel = [errorErrRel errorErrRelAux/datos_conocidos];

    % Norma "No aciertos"
    [m, n] = size(E);
    errorNoAciertosAux = 0;
    for i = 1:m
        for j = i:n
            if (E(i,j) > 0)
                if w(i) > w(j) && E(i,j) < 1
                    errorNoAciertosAux = errorNoAciertosAux + 1;
                elseif (w(i) == w(j) && E(i,j) ~= 1) || (w(i) ~= w(j) && E(i,j) == 1)
                    errorNoAciertosAux = errorNoAciertosAux + 0.5;
                end
            end
        end
    end
    errorNoAciertos = [errorNoAciertos errorNoAciertosAux/datos_conocidos];

    % Si solo nos pasan una matriz
    if (nargin == 2)
        break;
    end

    M = [M; E];
    W = [W; WAux];
    datos_conocidos_total = datos_conocidos_total + datos_conocidos;

end

if nargin > 2
    figure();
    l=compose("E%d",(1:nargin-1));
    % Pie
    tiledlayout(1, 3);
```

```
    ax1 = nexttile;
    pie(ax1, errorInf(1:end-1)./sum(errorInf(1:end-1)))
    legend(l)
    title('Error Inf')

    ax2 = nexttile;
    pie(ax2,errorFro(1:end-1)./sum(errorFro(1:end-1)))
    legend(l)
    title('Error Frobenius')

    ax3 = nexttile;
    pie(ax3,errorUno(1:end-1)./sum(errorUno(1:end-1)))
    legend(l)
    title('Error 1')
end

end
```

ic.m

```
function res = ic(M)
%Autovalor dominante
lambda = max(eig(M));
m = length(M);
%Calculo del indice de consistencia
res = (lambda - m) / (m - 1);
return
```