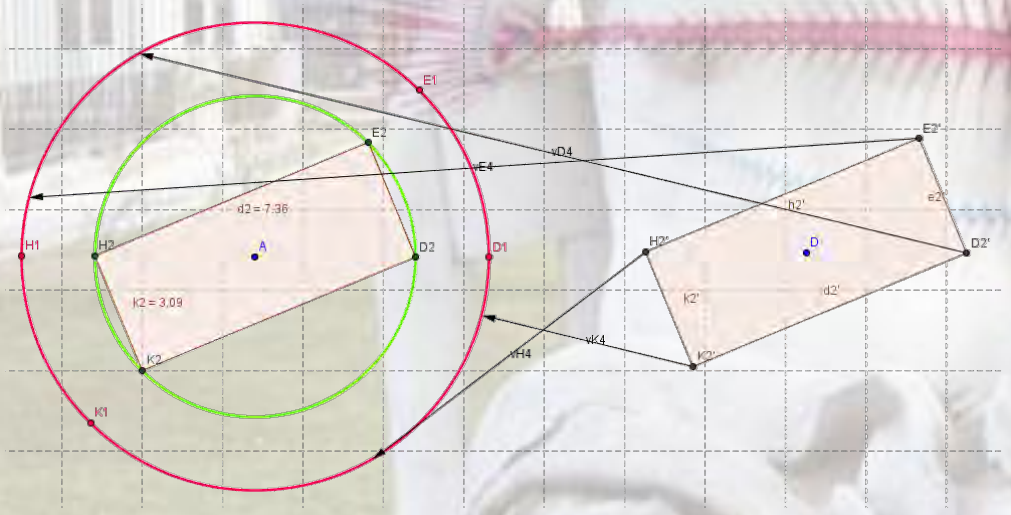
LABORATORIO DE ANATOMÍA ANIMAL

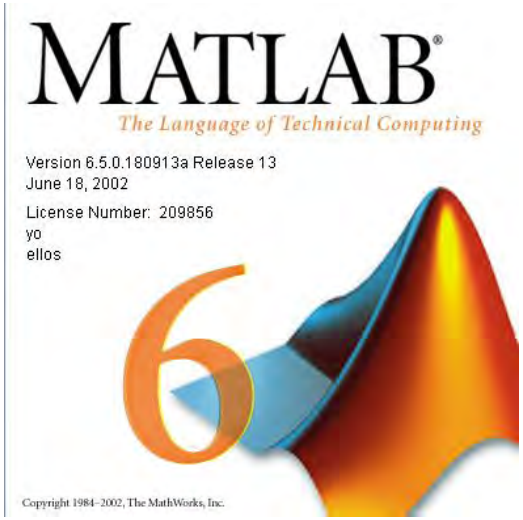
INGENIERIA INVERSA APLICADA A LA ANATOMÍA ANIMAL

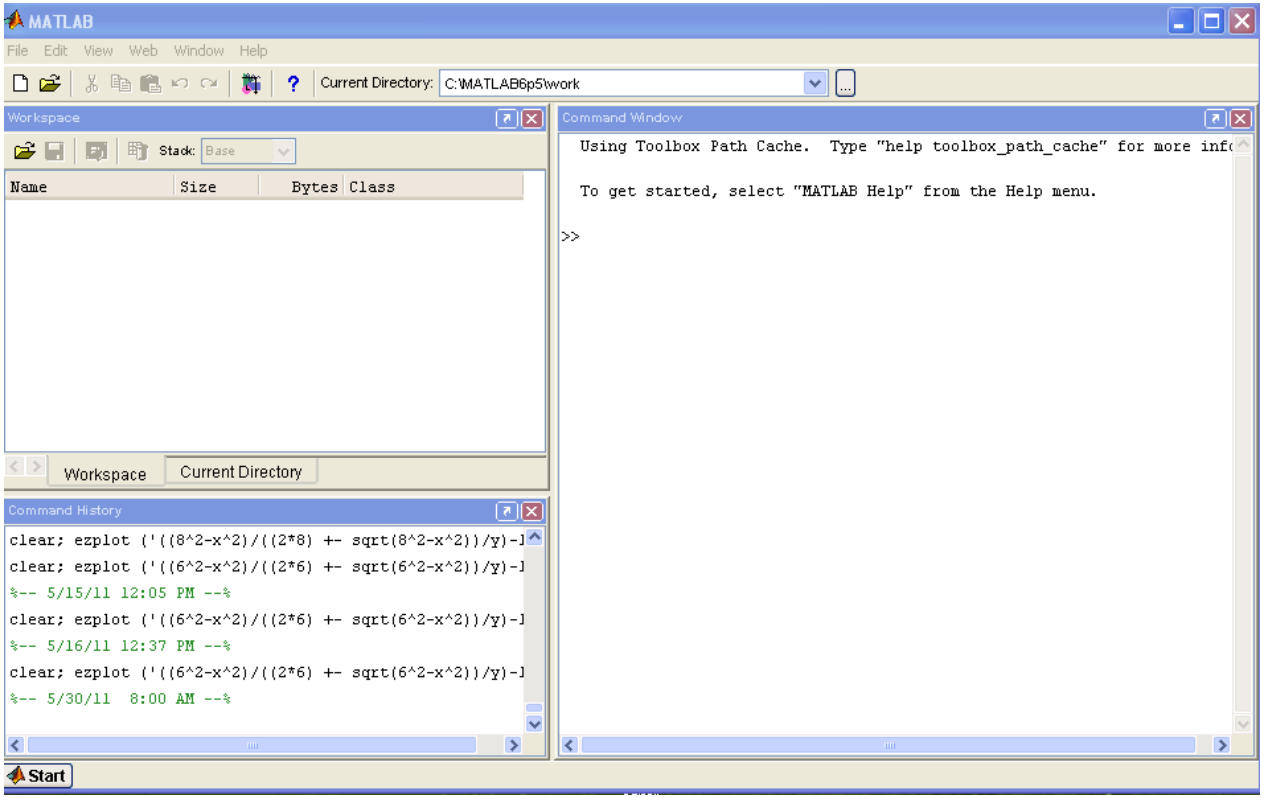
M O O C



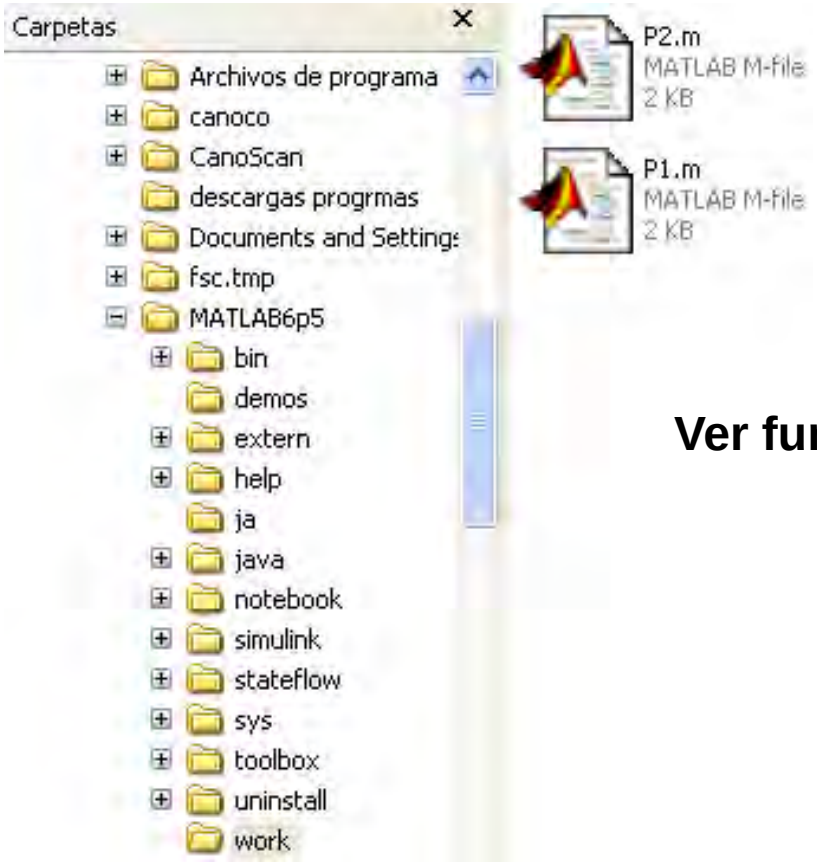
4.- Procrustes

Software


MATLAB®
The Language of Technical Computing
Version 6.5.0.180913a Release 13
June 18, 2002
License Number: 209856
yo
ellos
Copyright 1984-2002, The MathWorks, Inc.


MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work
Workspace
Stack: Base
Name Size Bytes Class
Workspace
Current Directory
Command History
clear; ezplot ('((8^2-x^2)/((2*8) +- sqrt(8^2-x^2))/y)-1
clear; ezplot ('((6^2-x^2)/((2*6) +- sqrt(6^2-x^2))/y)-1
%-- 5/15/11 12:05 PM --%
clear; ezplot ('((6^2-x^2)/((2*6) +- sqrt(6^2-x^2))/y)-1
%-- 5/16/11 12:37 PM --%
clear; ezplot ('((6^2-x^2)/((2*6) +- sqrt(6^2-x^2))/y)-1
%-- 5/30/11 8:00 AM --%
Command Window
Using Toolbox Path Cache. Type "help toolbox_path_cache" for more info
To get started, select "MATLAB Help" from the Help menu.
>>
Start

Carpeta work



Ver funciones en MOOC_004_work

Se llama matriz rectangular (A) de orden m x n a una tabla con m filas y n columnas. Cada elemento de una matriz tiene dos **subíndices**: **i** para las filas y **j** para las columnas. El primero i es constante en cada fila y el segundo j lo es en cada columna. La ***matriz cuadrada*** tiene el mismo numero de filas que de columnas. Cuando tiene una sola fila se llama **matriz fila** y cuando tiene una sola columna se llama **matriz columna.**

```
>> u = [2 4 5]

u =

     2     4     5

>> v = [2; 4; 5]

v =

     2
     4
     5

>> A = [1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

>>
```

matriz fila

matriz columna

```
>>  A = [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A = [1 2 3; 4 5 6; 7 8 9]'

A =

     1     4     7
     2     5     8
     3     6     9

>>
```

matriz cuadrada

La matriz traspuesta
Cambia filas por columnas

Dos matrices con el mismo numero de filas y de columnas pueden sumarse o restarse.

```
>> A = [1 2 3; 4 5 6; 7 8 9]'

A =

     1     4     7
     2     5     8
     3     6     9

>> B= [1 2 3; 4 5 6; 7 8 9]

B =

     1     2     3
     4     5     6
     7     8     9

>> C= A+B

C =

     2     6    10
     6    10    14
    10    14    18

>>
```

```
>> D=A-B

D =

     0     2     4
    -2     0     2
    -4    -2     0

>>
```

El producto matricial no es acumulativo.

```
>> A = [1 2 3; 4 5 6; 7 8 9]'

A =

     1     4     7
     2     5     8
     3     6     9

>> B= [1 2 3; 4 5 6; 7 8 9]

B =

     1     2     3
     4     5     6
     7     8     9
```

```
>> C=A*B

C =

    66    78    90
    78    93   108
    90   108   126

>> D=B*A

D =

    14    32    50
    32    77   122
    50   122   194

>>
```

A =

2	2	2
3	3	3

b =

2
2
2

>> A*b

12
18

2x2 +2x2+2x2=12
2x3+2x3+2x3=18

Hay que tener en cuenta que, para obtener un producto, el número de columnas de la primera matriz debe ser igual al número de filas de la segunda.

A =

2	2	2
3	3	3

b =

2	1
2	1
2	1

>> A*b

12	6
18	9

>> b*A

7	7	7
7	7	7
7	7	7

2x2+3x1

>> A*10

20	20	20
30	30	30

>> b*A/7

1	1	1
1	1	1
1	1	1

Cuando tres matrices son de órdenes m x n, n x p, p x q, puede obtenerse el producto de las tres

```
>> clear;
>> A= [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> B=[1 2; 2 3 ;3 4]

B =

     1     2
     2     3
     3     4

>> C=[1 2; 2 3]

C =

     1     2
     2     3

>> D=A*B*C

D =

    54    88
   126   205
   198   322

>>
```

La traspuesta del producto de dos matrices es igual al producto de la traspuesta de la segunda matriz por la traspuesta de la primera.

```
>>  A= [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> B=[1 2; 2 3 ;3 4]

B =

     1     2
     2     3
     3     4

>> C=A*B

C =

    14    20
    32    47
    50    74
```

```
>> D=C'

D =

    14    32    50
    20    47    74

>> E=B'*A'

E =

    14    32    50
    20    47    74
```

Multiplicando una matriz por la **matriz unidad** se obtiene la misma matriz.

```
>> A= [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1

>> A*eye(3)

ans =

     1     2     3
     4     5     6
     7     8     9
```

matriz unidad

```
>> eye(1)

ans =

     1

>> A*eye(1)

ans =

     1     2     3
     4     5     6
     7     8     9

>> eye(2)

ans =

     1     0
     0     1

>> A*eye(2)
??? Error using ==> *
Inner matrix dimensions must agree.
```

Una matriz multiplicada por un **escalar** K se obtienemultiplicando cada uno de sus elementos por ese escalar.

```
>> A= [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A*2

ans =

     2     4     6
     8    10    12
    14    16    18
```

Para una matriz cuadrada A[n,n], el **determinante** de A, abreviado det(A), es un escalar definido como la suma de n! términos involucrando el producto de n elementos de la matriz, cada uno proveniente exactamente de una fila y columna diferente. Además, cada término de la suma está multiplicado por −1 ó +1 dependiendo del número de permutaciones del orden de las columnas que contenga.

```
>> A= [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> det(A)

ans =

     0
```

```
>> B= [1 4 3; 8 5 6; 7 8 1]

B =

     1     4     3
     8     5     6
     7     8     1

>> det(B)

ans =

   180
```

$$A = (a_{11})$$

El valor del determinante es igual al único termino de la matriz:

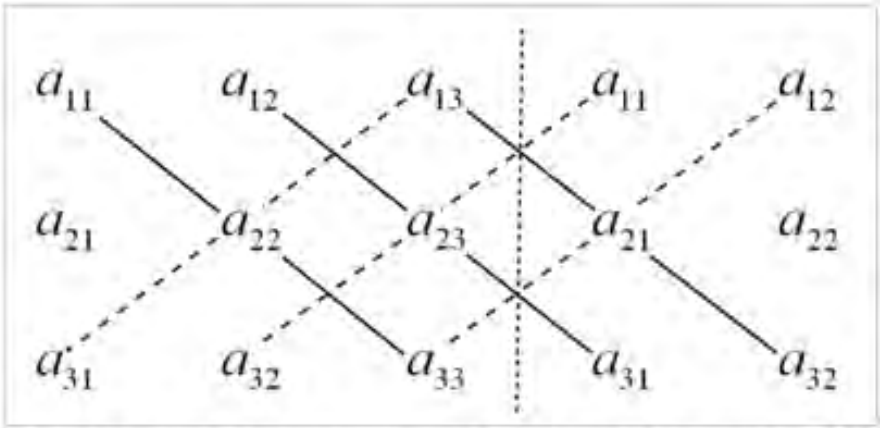
$$\det A = \det (a_{11}) = |a_{11}| = a_{1,1}$$

Los determinantes de una matriz de orden 2:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

se calculan con la siguiente fórmula:

$$\det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}$$



La *regla de Sarrus*: las diagonales continuas se suman y las diagonales en trazos se restan.

Dada una matriz de orden 3:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

En determinante de orden 3 se calcula mediante la regla de Sarrus:

$$\det \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} =$$

$$= a_{1,1}a_{2,2}a_{3,3} + a_{1,2}a_{2,3}a_{3,1} + a_{1,3}a_{2,1}a_{3,2} - (a_{1,3}a_{2,2}a_{3,1} + a_{1,2}a_{2,1}a_{3,3} + a_{1,1}a_{2,3}a_{3,2})$$

Si (A) x (B) = (C), los determinantes de las matrices también cumplen la condición Det(A) x Det(B) = Det(C).

```
>> det(A)

ans =

     0

>> det(B)

ans =

   180

>> A*B

ans =

    38    38    18
    86    89    48
   134   140    78

>> det(A*B)

ans =

     0
```

```
>> A= [10 4 3; 8 5 8; 7 8 5]

A =

    10     4     3
     8     5     8
     7     8     5

>> B= [1 4 3; 8 5 6; 7 8 1]

B =

     1     4     3
     8     5     6
     7     8     1

>> det(A)

ans =

  -239

>> det(B)

ans =

   180
```

```
>> det(A)*det(B)

ans =

      -43020

>> C=A*B

C =

    63    84    57
   104   121    62
   106   108    74

>> det(C)

ans =

      -43020
```

Se denomina **matriz inversa** de una matriz cuadrada (A) a otra matriz $(A)^{-1}$ que multiplicada por (A) es la matriz unidad (U).

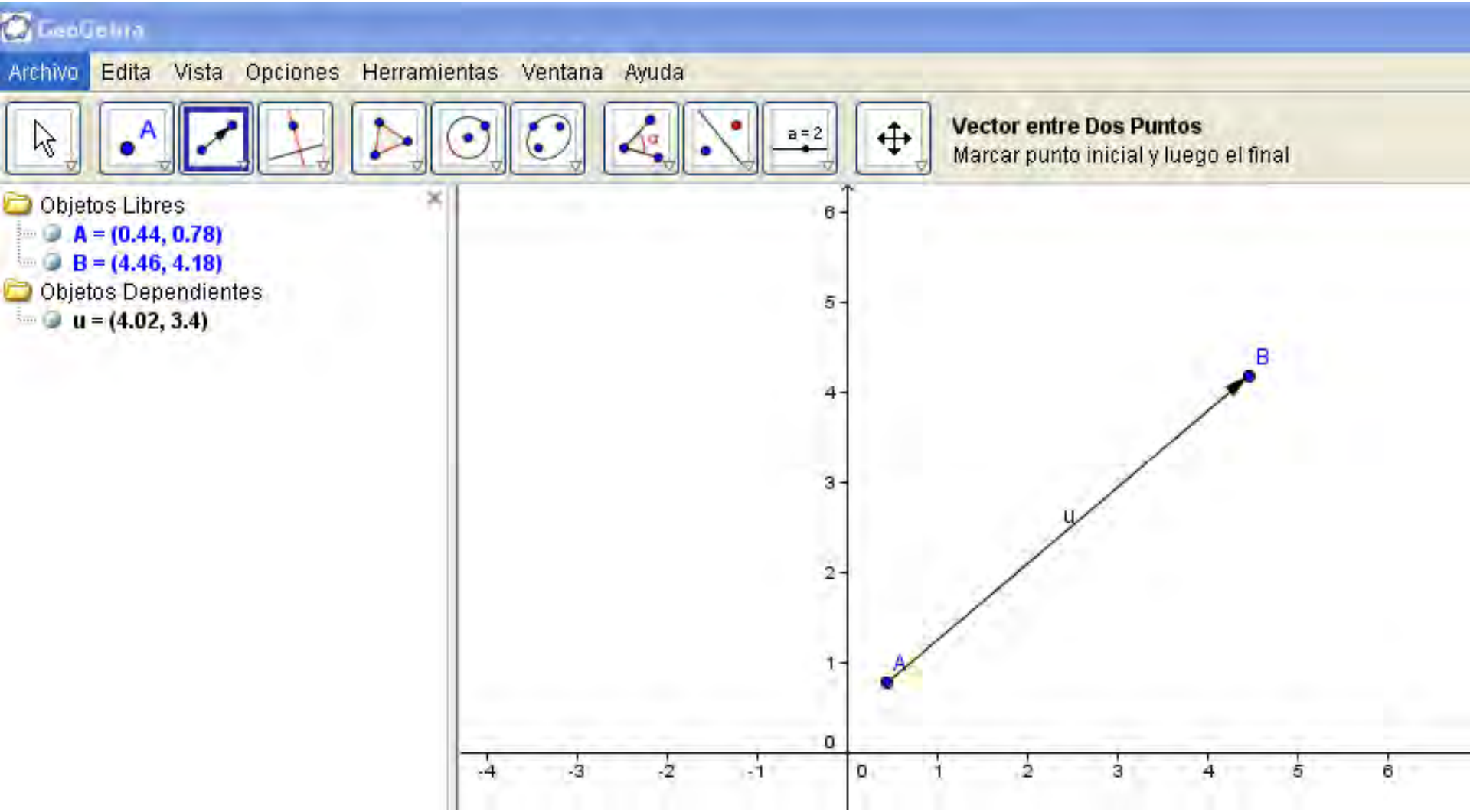
```
>> A= [10 4 3; 8 5 8; 7 8 5]

A =

    10     4     3
     8     5     8
     7     8     5

>> inv(A)

ans =

    0.1632   -0.0167   -0.0711
   -0.0669   -0.1213    0.2343
   -0.1213    0.2176   -0.0753

>> A*inv(A)

ans =

    1.0000   -0.0000         0
    0.0000    1.0000    0.0000
    0.0000   -0.0000    1.0000
```

Aplicando el producto de determinantes ha de verificar D(A) x D(A) $^{-1}$ =1. Si una matriz tiene un determinante distinto de 0 se llama **matriz regular**. Por tanto la inversa de una matriz cuando existe es única.

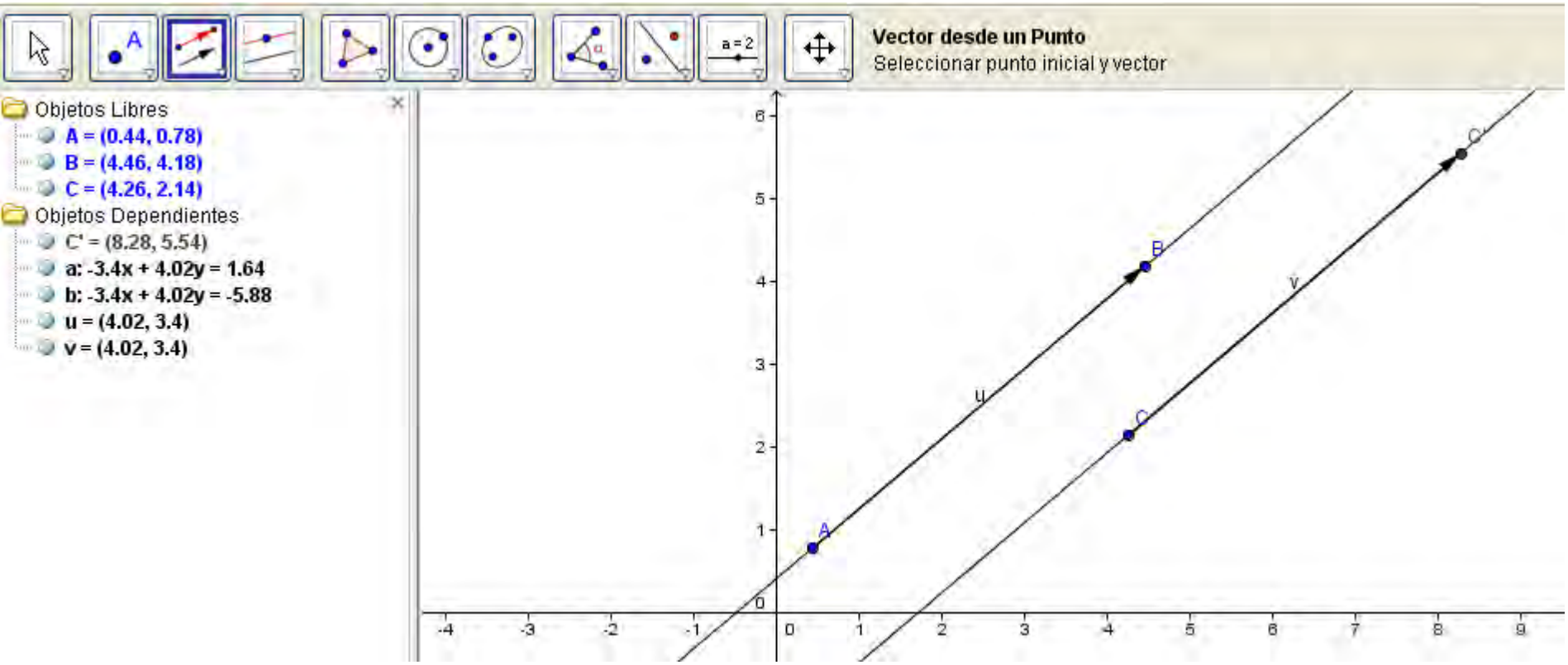
```
>> A= [10 4 3; 8 5 8; 7 8 5]

A =

    10     4     3
     8     5     8
     7     8     5

>> B= inv(A)

B =

    0.1632   -0.0167   -0.0711
   -0.0669   -0.1213    0.2343
   -0.1213    0.2176   -0.0753

>> A*B

ans =

    1.0000   -0.0000         0
    0.0000    1.0000    0.0000
    0.0000   -0.0000    1.0000
```

```
>> det(A)

ans =

   -239

>> det(B)

ans =

   -0.0042

>> det(A)*det(B)

ans =

    1.0000
```

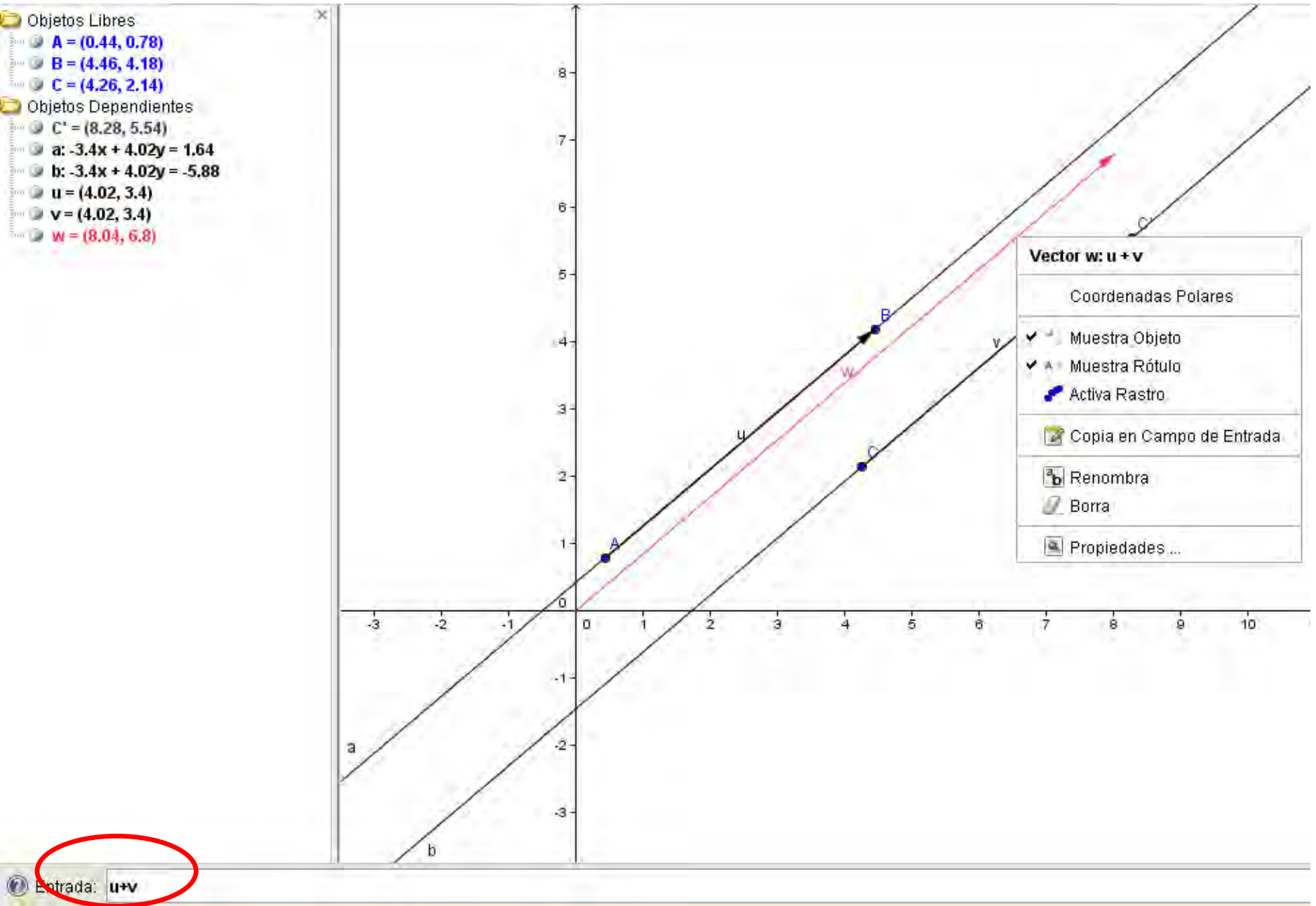
Se define como **vector** a un segmento AB que tiene una longitud o **módulo** determinado, una **dirección** que coincide con la recta que lo contiene y un **sentido** representado por el movimiento de un punto que se desplaza de A a B, llamándose a A **origen** y a B **extremo** del vector. Vector unitario o **versor** es el que tiene de longitud la unidad.



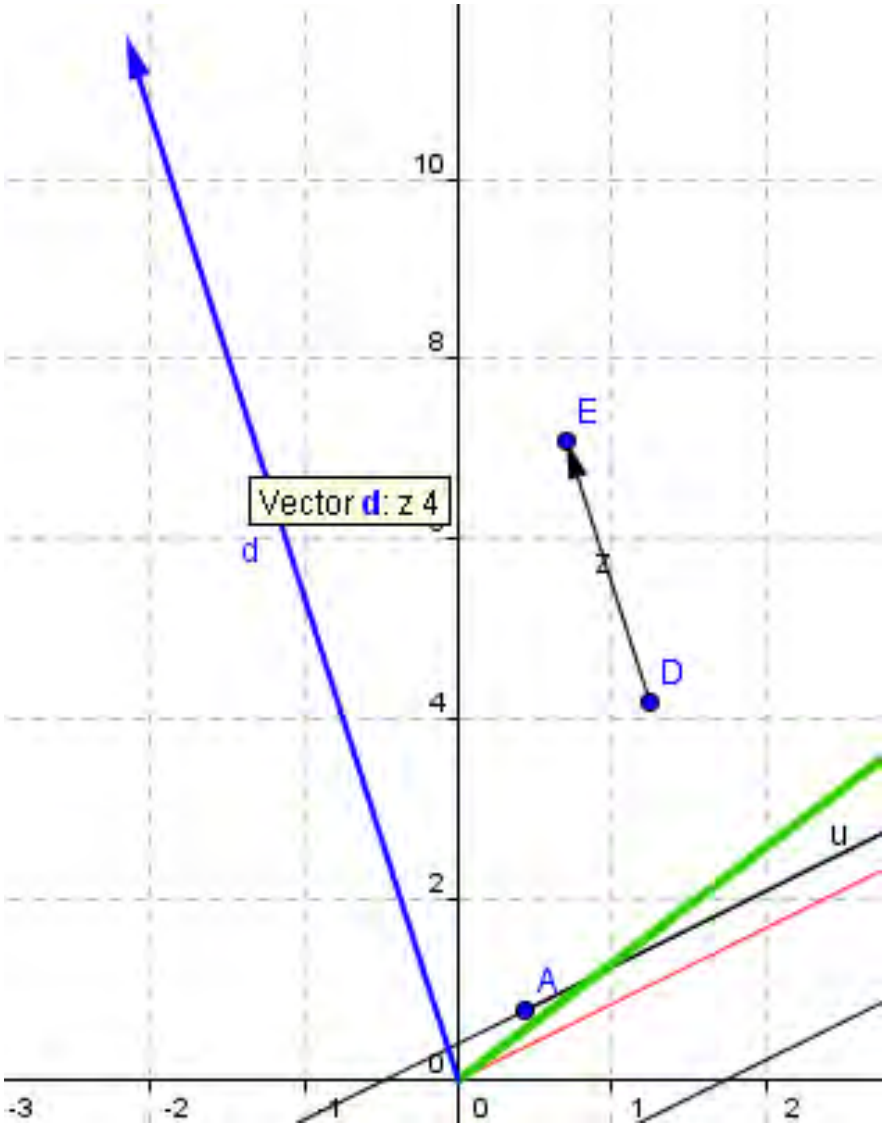
Dos vectores situados en dos rectas distintas tienen la misma dirección si ambas rectas son paralelas. Dos vectores que tienen la misma dirección, el mismo sentido e igual módulo se llaman **vectores libres iguales**.

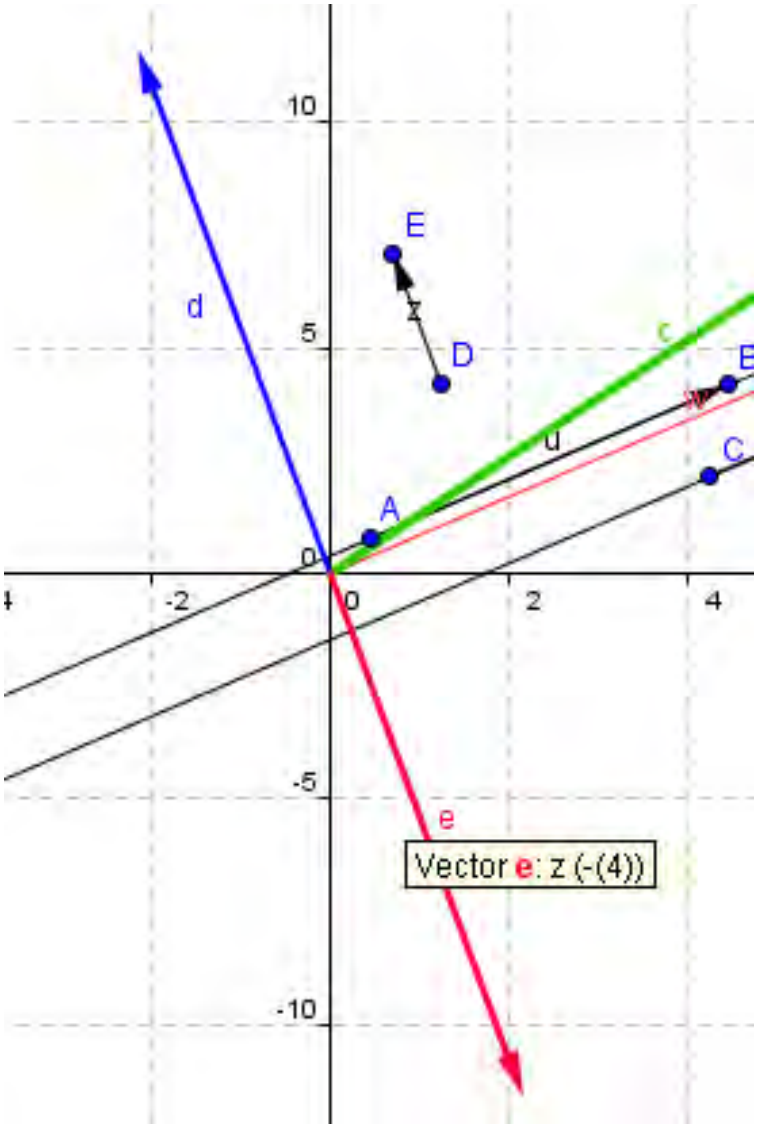


Suma de vectores con una misma dirección: el vector suma w = u+v

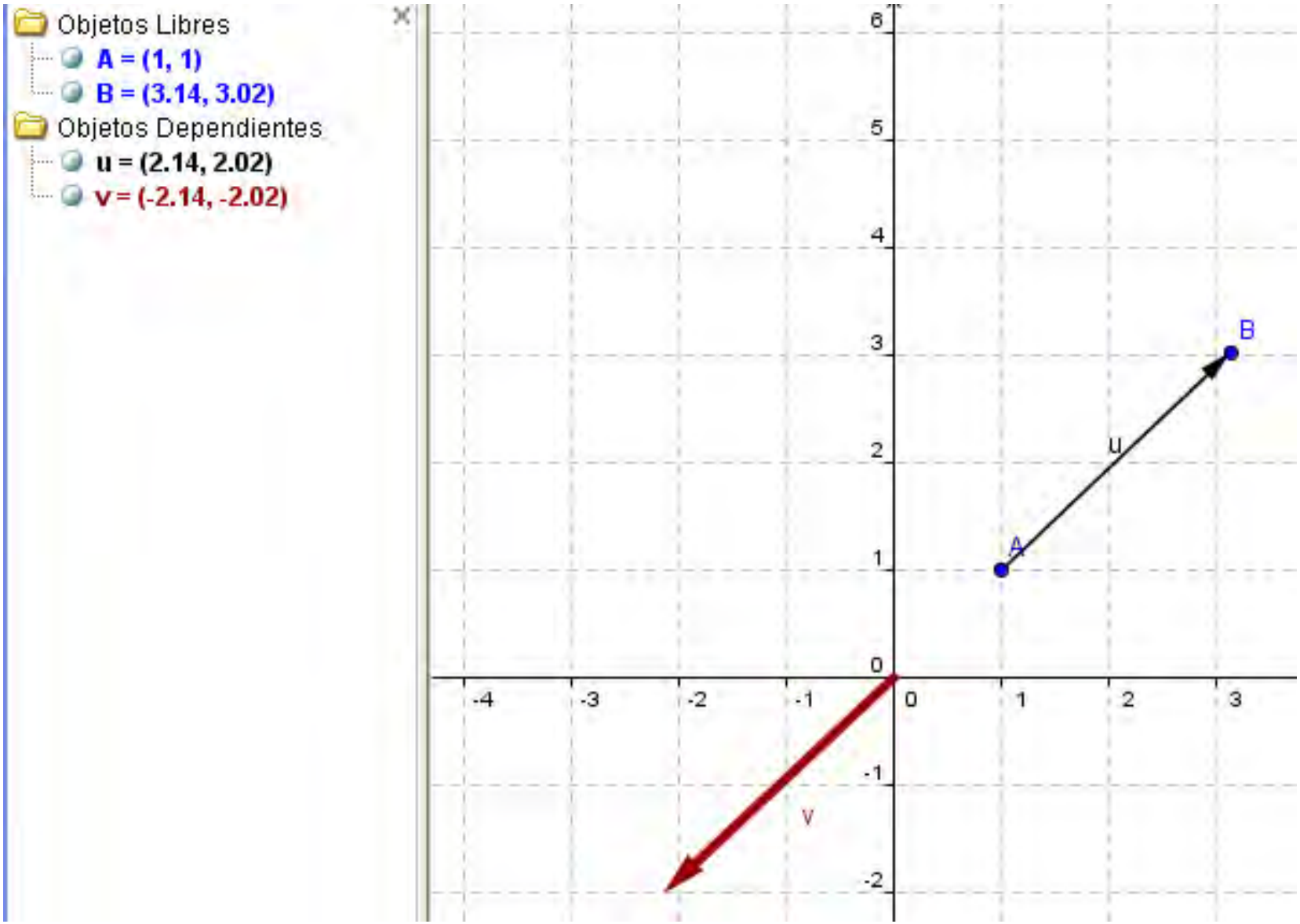


El producto de un vector por un número real n es otro vector de la misma dirección y sentido si n es positivo o de sentido contrario si n es negativo. En ambos casos el módulo del vector resultante es el valor absoluto del modulo del vector multiplicado por n.

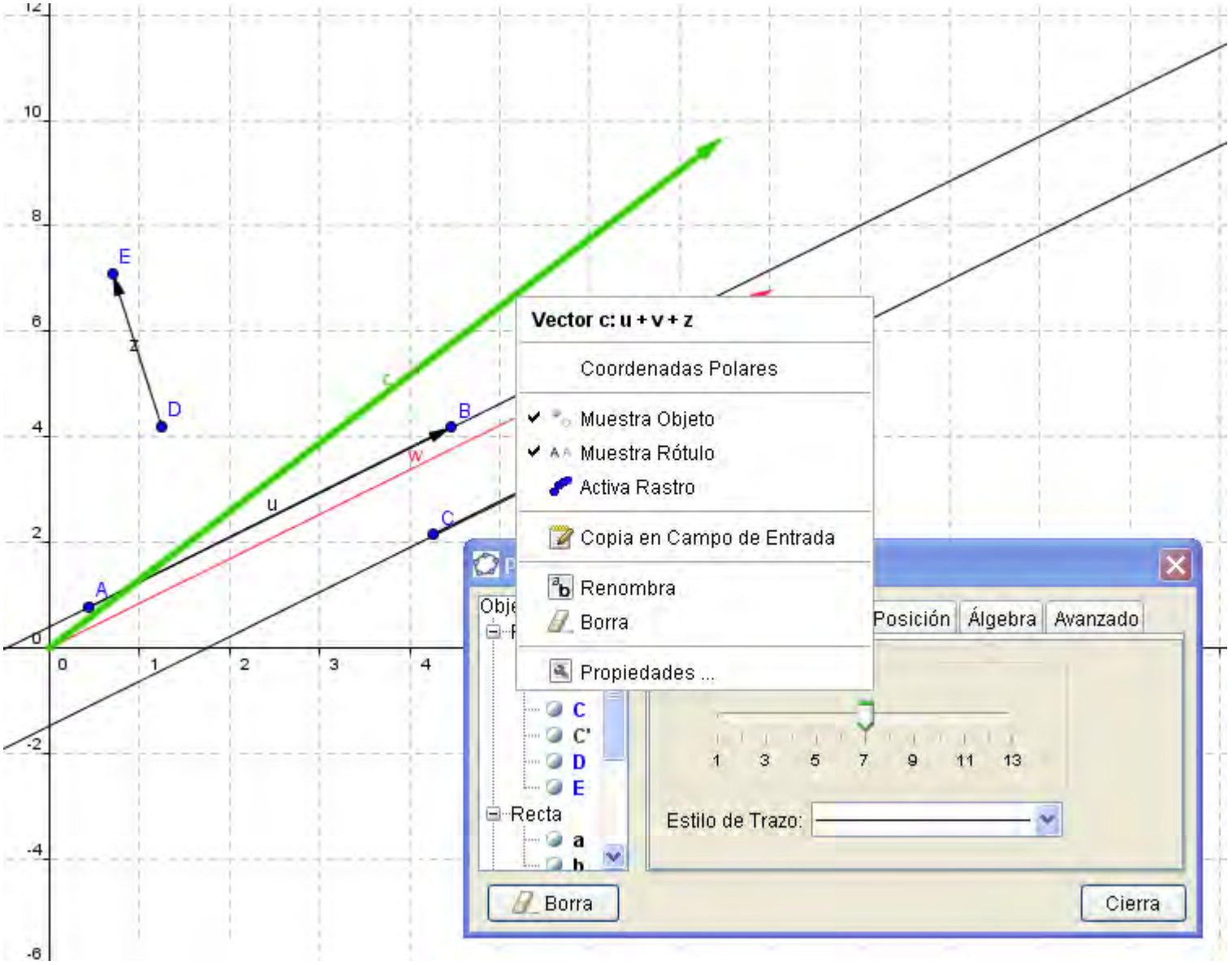




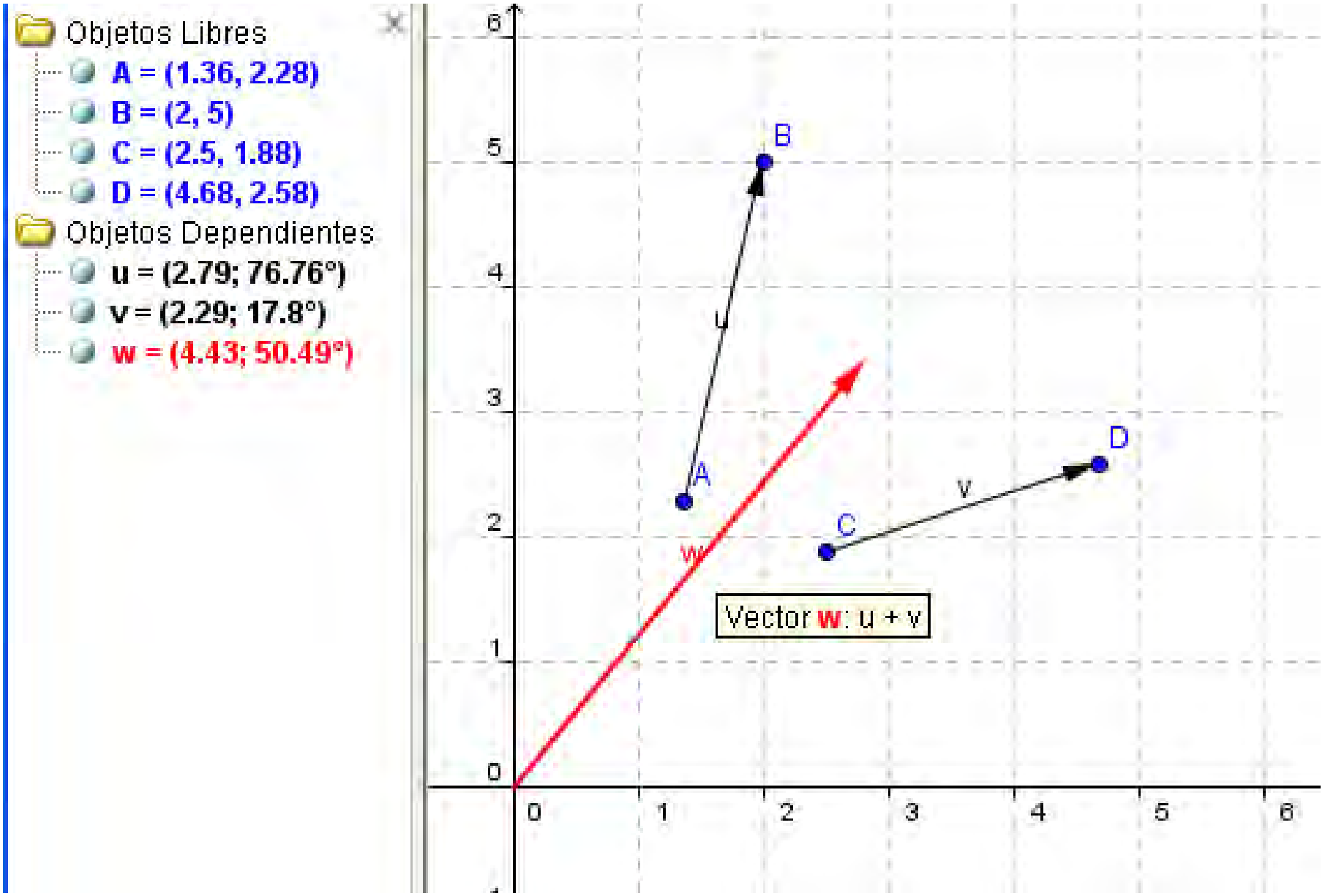
El producto de un vector por -1 se denomina **vector opuesto** (v es el vector opuesto a u).

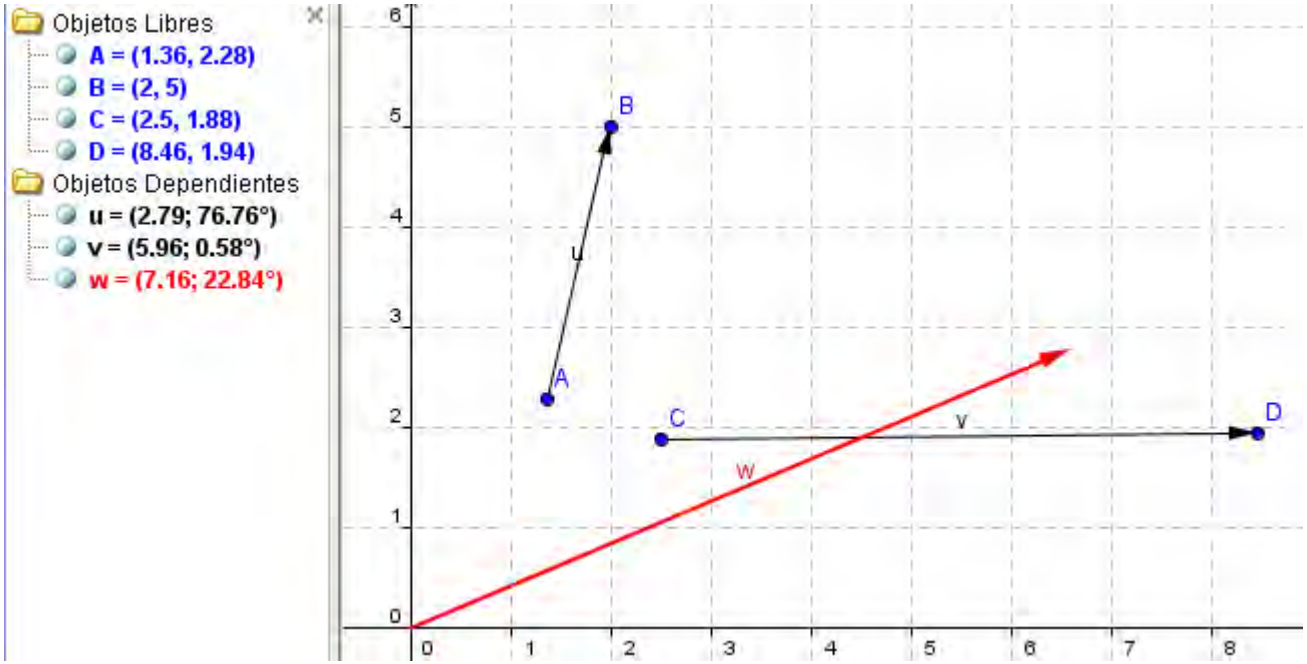


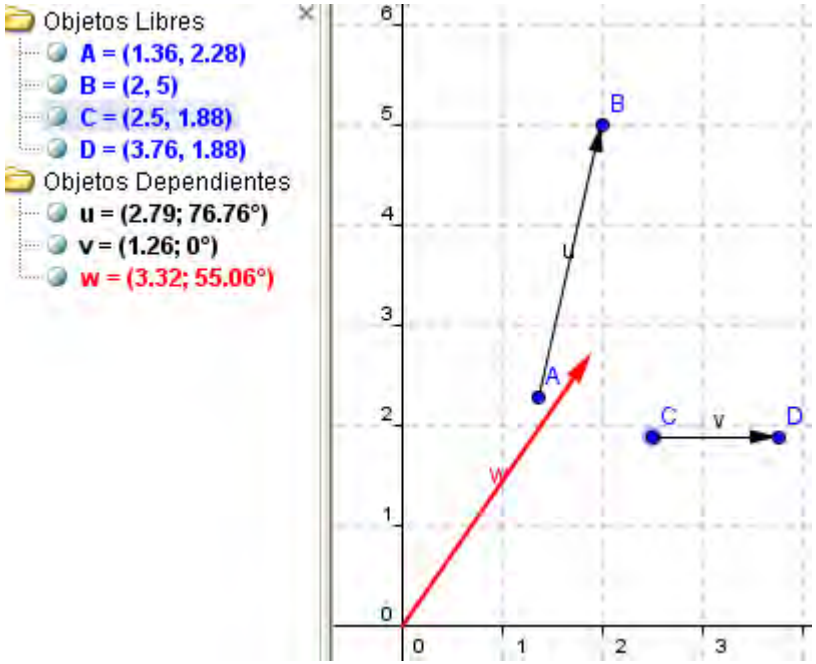
La suma de vectores es uniforme (si r=l y w=z, r+w = l+z), es conmutativa y es asociativa.



Suma de vectores de distinta dirección




Objetos Libres
A = (1.36, 2.28)
B = (2, 5)
C = (2.5, 1.88)
D = (8.46, 1.94)
Objetos Dependientes
u = (2.79; 76.76°)
v = (5.96; 0.58°)
w = (7.16; 22.84°)
A
B
C
D
u
v
w


Objetos Libres
A = (1.36, 2.28)
B = (2, 5)
C = (2.5, 1.88)
D = (3.76, 1.88)
Objetos Dependientes
u = (2.79; 76.76°)
v = (1.26; 0°)
w = (3.32; 55.06°)
A
B
C
D
u
v
w

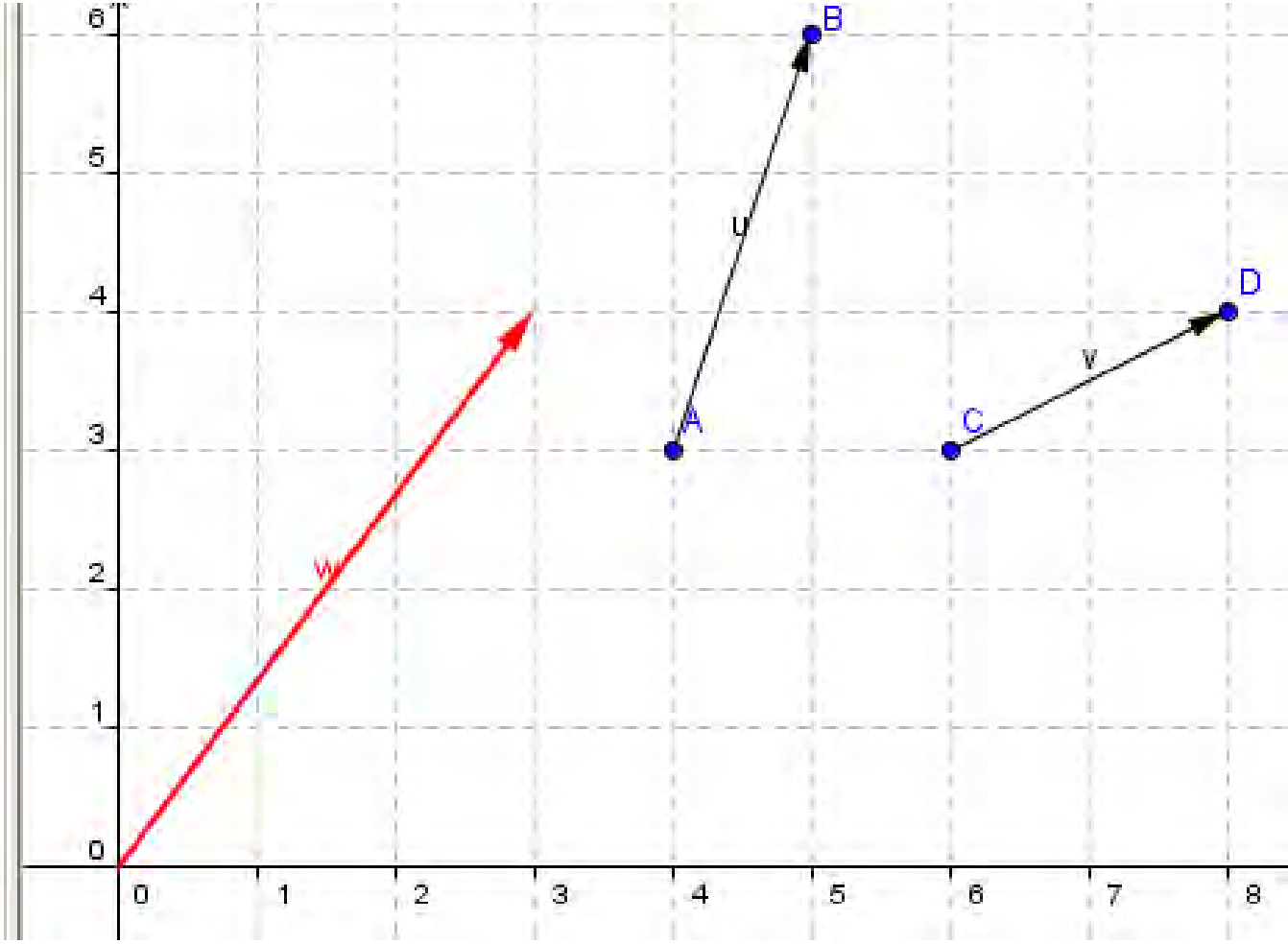
Valores del módulo y del ángulo:

Coordenadas cartesianas

- Objetos Libres
  - A = (4, 3)
  - B = (5, 6)
  - C = (6, 3)
  - D = (8, 4)
- Objetos Dependientes
  - u = (1, 3)
  - v = (2, 1)
  - w = (3, 4)

Coordenadas polares

- Objetos Libres
  - A = (4, 3)
  - B = (5, 6)
  - C = (6, 3)
  - D = (8, 4)
- Objetos Dependientes
  - u = (3.16; 71.57°)
  - v = (2.24; 26.57°)
  - w = (5; 53.13°)



Coordenadas cartesianas

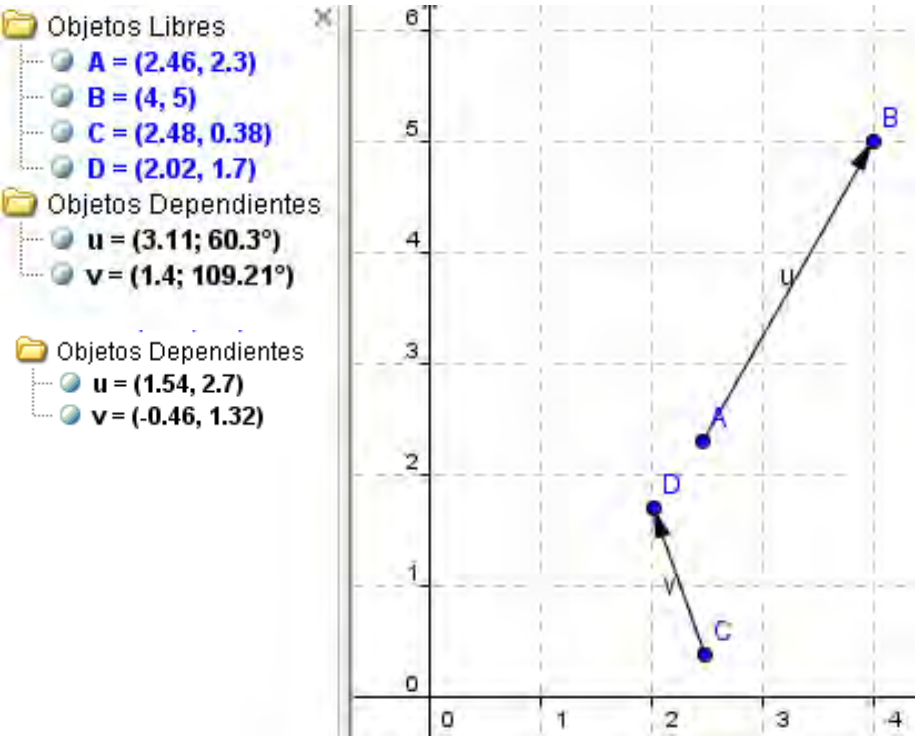
```
>> A= [4;3]

A =

     4
     3

>> B=[5;6]

B =

     5
     6

>> u=B-A

u =

     1
     3
```

```
>> C=[6;3]

C =

     6
     3

>> D=[8;4]

D =

     8
     4

>> v=D-C

v =

     2
     1
```

```
>> % Calcula distancias utilizando la norma euclidiana
[n, N]=size(A);
[n, c]=size(B);
for i=1:c
for j=1:N
tempp=0;
for k=1:n
temp(k)=(A(k,j)-B(k,i))^2;
tempp=tempp+temp(k);
end
d(i,j)=sqrt(tempp);
end
end
>> d

d =

    3.1623
```

```
>> clear; C=[6;3]; D=[8;4];
>> [n, N]=size(C);
[n, c]=size(D);
for i=1:c
for j=1:N
tempp=0;
for k=1:n
temp(k)=(C(k,j)-D(k,i))^2;
tempp=tempp+temp(k);
end
d(i,j)=sqrt(tempp);
end
end
>> d

d =

    2.2361
```

Cálculo multiple de vectores




```
>> clear; AyC=[2.46 2.48;2.3 0.38]; ByD=[4 2.02;5  1.7];
>> [n, N]=size(AyC);
[n, c]=size(ByD);
for i=1:c
for j=1:N
tempp=0;
for k=1:n
temp(k)=(AyC(k,j)-ByD(k,i))^2;
tempp=tempp+temp(k);
end
d(i,j)=sqrt(tempp);
end
end
>> d

d =

    3.1083    4.8636
    0.7440    1.3979
```

Distancias euclideas			
Vector		**Origen**	
		A	**C**
Extremo	**B**	**3.1083**	**4.8636**
	D	**0.7440**	**1.3979**

```
>> diag(d)

ans =

    3.1083
    1.3979
```

Módulo de los vectores

u

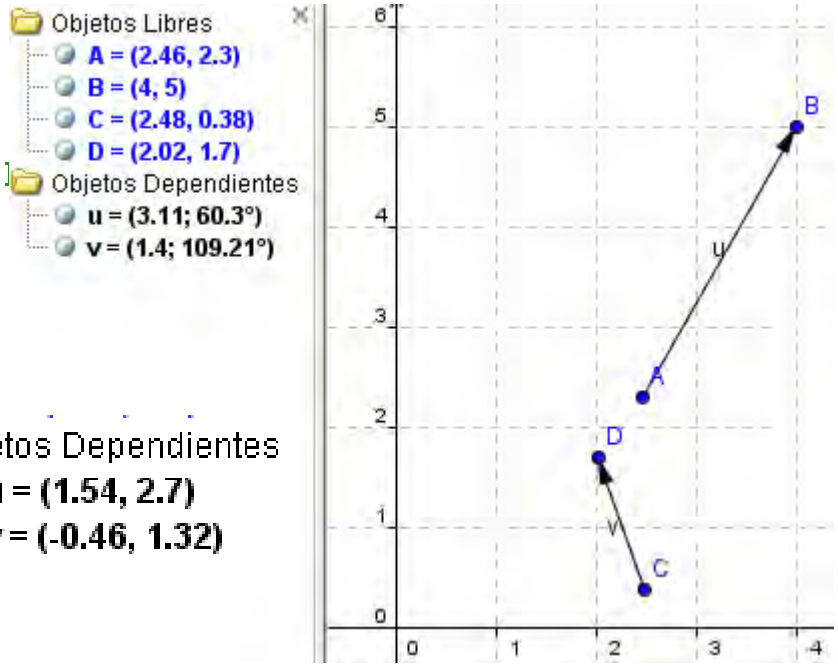
v

```
>> clear; AyC=[2.46 2.48;2.3 0.38]; ByD=[4 2.02;5  1.7];catetos=ByD-AyC;
 angulo=atan([catetos(2,:)]./catetos(1,:))*180/pi;

for i=1:length(angulo)
n=angulo(1,i);
if n < 1, angulo(1,i)=n+180;
end;
end;

angulos=angulo'; %pasar la fila a columna.
>> angulos

angulos =

   60.3008
  109.2127
```

Ángulo de los vectores

u

v




```
>> polares=[modulos angulos]

polares =

    3.1083   60.3008
    1.3979  109.2127
```

```
>> total=[polares catetos']

total =

    3.1083   60.3008    1.5400    2.7000
    1.3979  109.2127   -0.4600    1.3200
```

```
>> total=[polares catetos' AyC']

total =
Columnas:                                                          Vectores
      1         2         3         4         5         6                filas
 3.1083   60.3008    1.5400    2.7000    2.4600    2.3000    <--  u       1
 1.3979  109.2127   -0.4600    1.3200    2.4800    0.3800    <--  v       2
                                                                         3
>>                                                                       etc..
```

módulo

ángulo

catetos

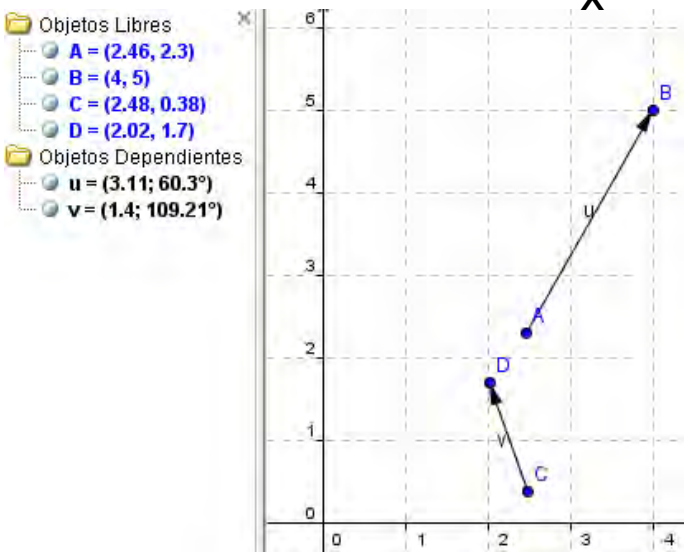
x

y

punto de origen

x

y



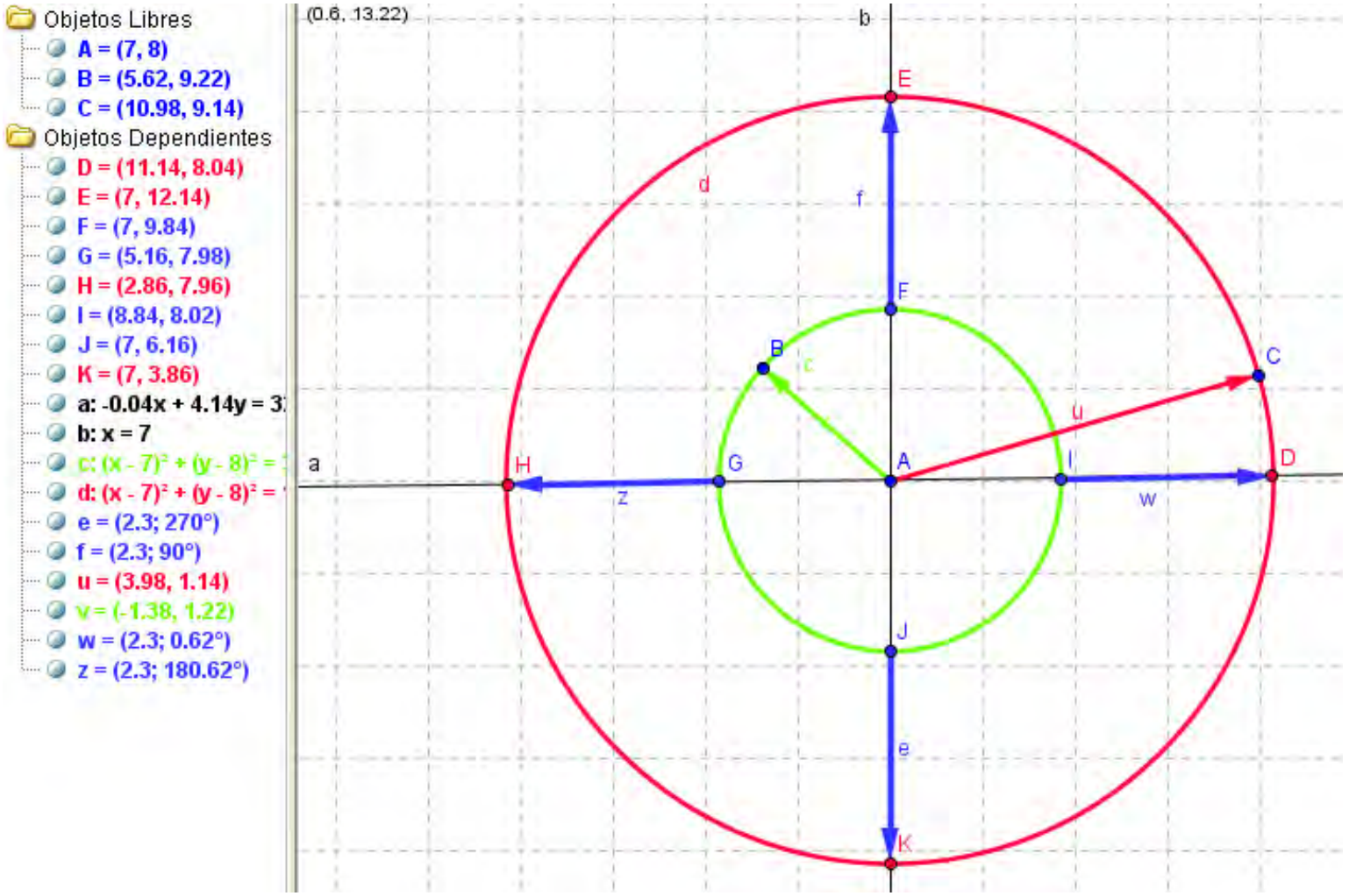
Matriz 1

Código: matriz_4.txt

```
%Cuadrado
clear; origen=[12.61  10.89 0.99  2.71 ;8.82  12.97  8.86  4.71];
extremo=[3.91  1.17  9.69  12.43 ;13.88 10.3  3.8 7.38];%destino
%Circunferencia
clear; origen=[8.84 7  5.16  7;8.02 9.84 7.98 6.16]; extremo=[11.14 7
2.86  7;8.04  12.14 7.96 3.86];
catetos=extremo-origen;
cateto=catetos
for i=1:length(catetos);
 n=catetos(1,i);
 nn=catetos(2,i);
 if n == 0  , catetos(1,i)=catetos(1,i)+0.0001;end;
 if nn == 0  , catetos(2,i)=catetos(2,i)+0.0001 ;end;% para evitar:  Warning: Divide by zero.
end;
 angulo=atan([catetos(2,:)]./catetos(1,:))*180/pi;
for i=1:length(catetos)
 n=catetos(1,i);
 nn=catetos(2,i);
  if  n < 0
    angulo(1,i)=angulo(1,i)+180;
  elseif n > 0 & nn < 0
     angulo(1,i)=angulo(1,i)+360;
  else
    angulo(1,i)=angulo(1,i);
  end
end;
angulos=angulo'; %pasar la fila a columna. Matriz ángulos.
[filas, colorigen]=size(origen);%matriz origen
[filas, colextremo]=size(extremo);%matriz extremo
for i=1:colextremo
for j=1:colorigen
tempp=0;
for k=1:filas
temp(k)=(origen(k,j)-extremo(k,i))^2;
tempp=tempp+temp(k);
end
modulo(i,j)=sqrt(tempp);
end
end
distancias=modulo
modulos=diag(modulo); % matriz módulos;
polares=[modulos angulos];
total=[polares cateto' origen']
```

El tamaño diferente con la misma forma geométrica es una semejanza. El grupo de las semejanzas sirve de base a la **geometría métrica**.

Escala



w	2.3001	0.4982	2.3000	0.0200	8.8400	8.0200
f	2.3000	89.9975	0	2.3000	7.0000	9.8400
z	2.3001	180.4982	-2.3000	-0.0200	5.1600	7.9800
e	2.3000	269.9975	0	-2.3000	7.0000	6.1600

Escala

distancias = La diagonal en rojo son las distancias euclideas
de los vectores que producen el cambio de escala.

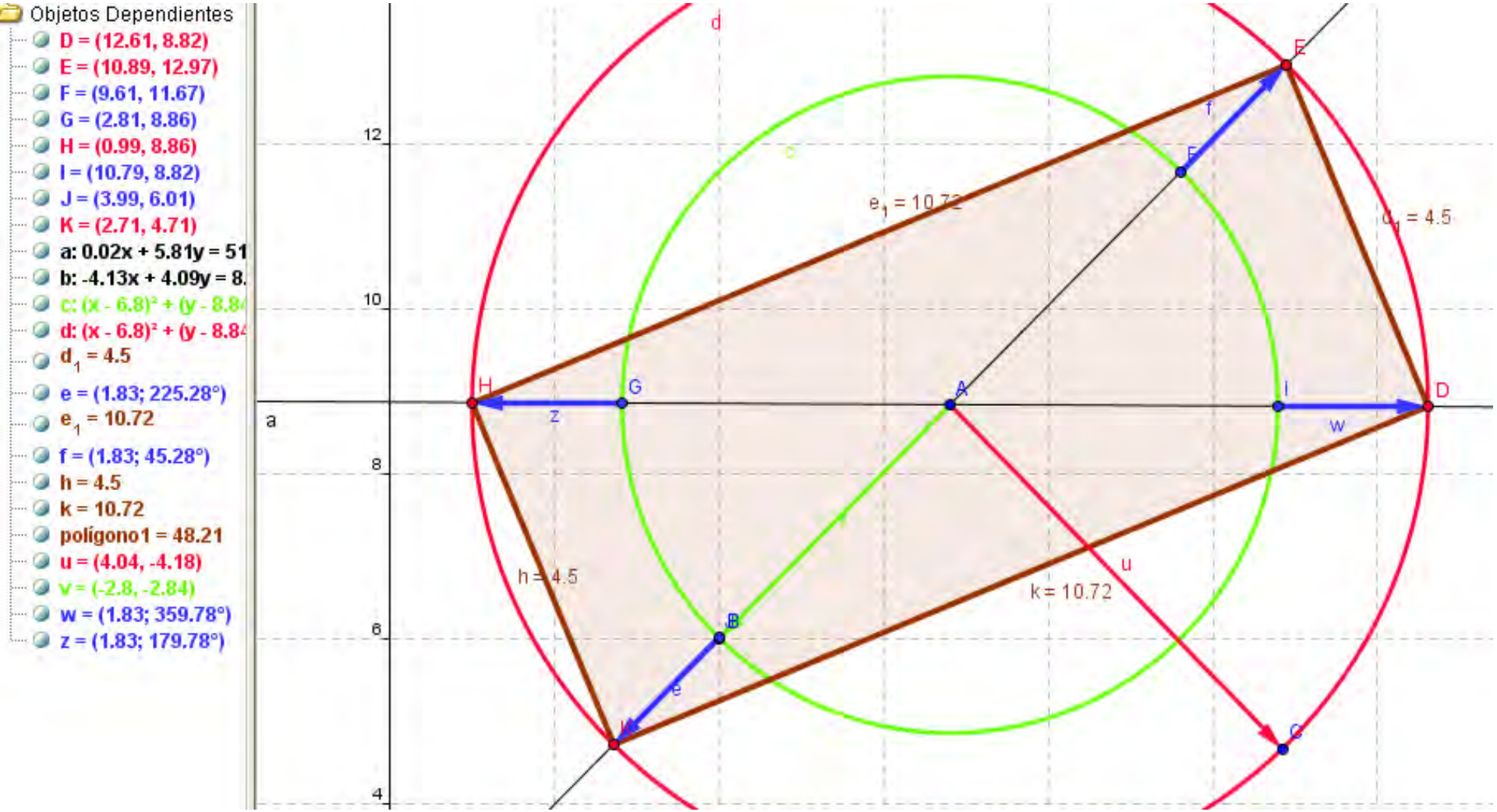
	(w) I	(f) F	(z) G	(e) J
(w) D	2.3001	4.5144	5.9803	4.5469
(f) E	4.5122	2.3000	4.5488	5.9800
(z) H	5.9803	4.5469	2.3001	4.5144
(e) K	4.5488	5.9800	4.5122	2.3000

La mayor distancia (FK) menos la menor (FE) es el radio de la circunferencia origen, mientras que su suma es el radio de la circunferencia destino. La escala es el cociente.

```
>> minimos =min(distancias)
   2.3001    2.3000    2.3001    2.3000
>> maximos =max(distancias)
   5.9803    5.9800    5.9803    5.9800
>> maximos+minimos
8.2804    8.2800    8.2804    8.2800

>> maximos-minimos
   3.6802    3.6800    3.6802    3.6800

>> escala=(maximos+minimos)./(maximos-minimos)
   2.2500    2.2500    2.2500    2.2500

>> [max(escala) min(escala)]
   2.2500    2.2500
```

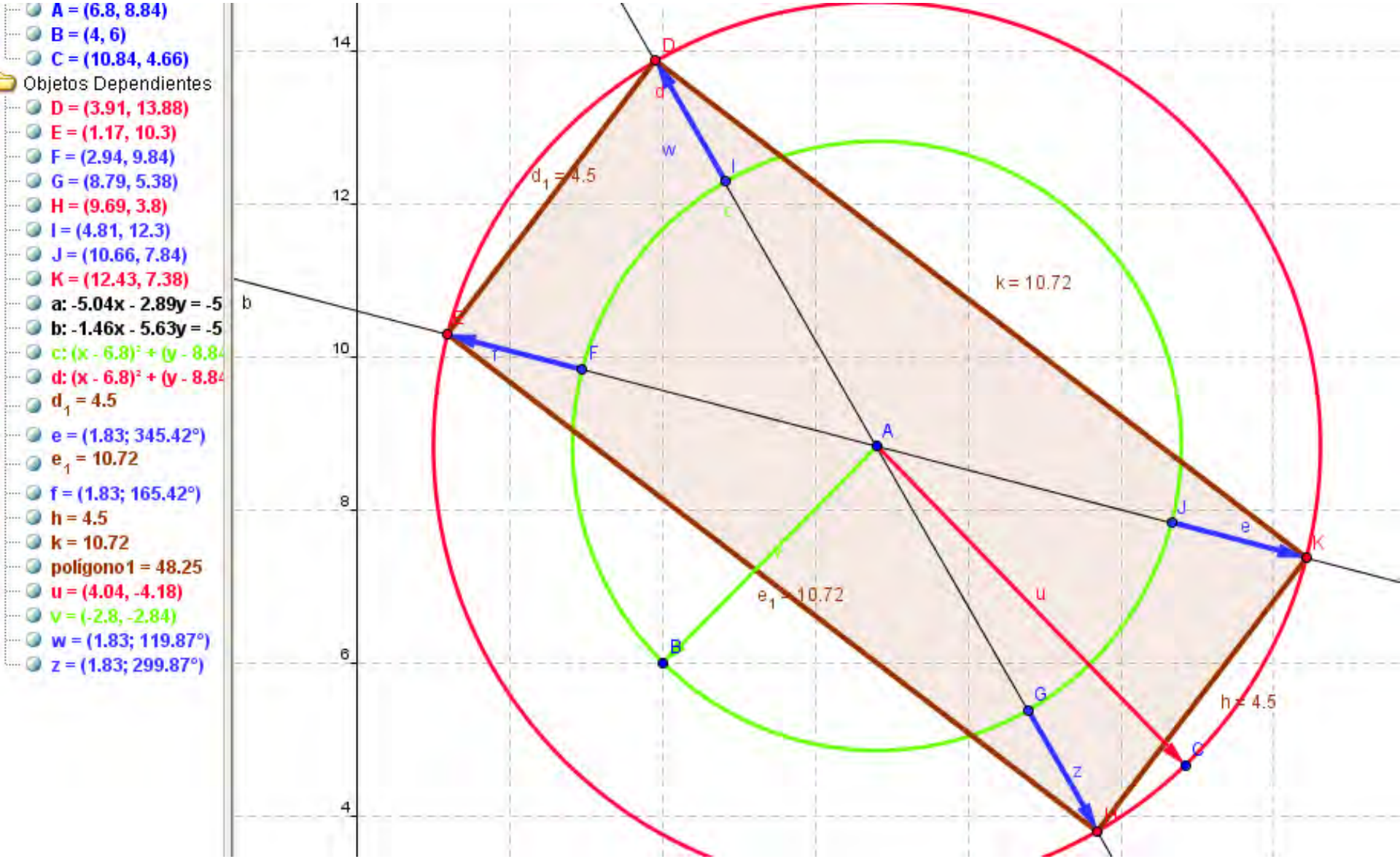
Rotación

origen P=[12.61 10.89 0.99 2.71 ;8.82 12.97 8.86 4.71]



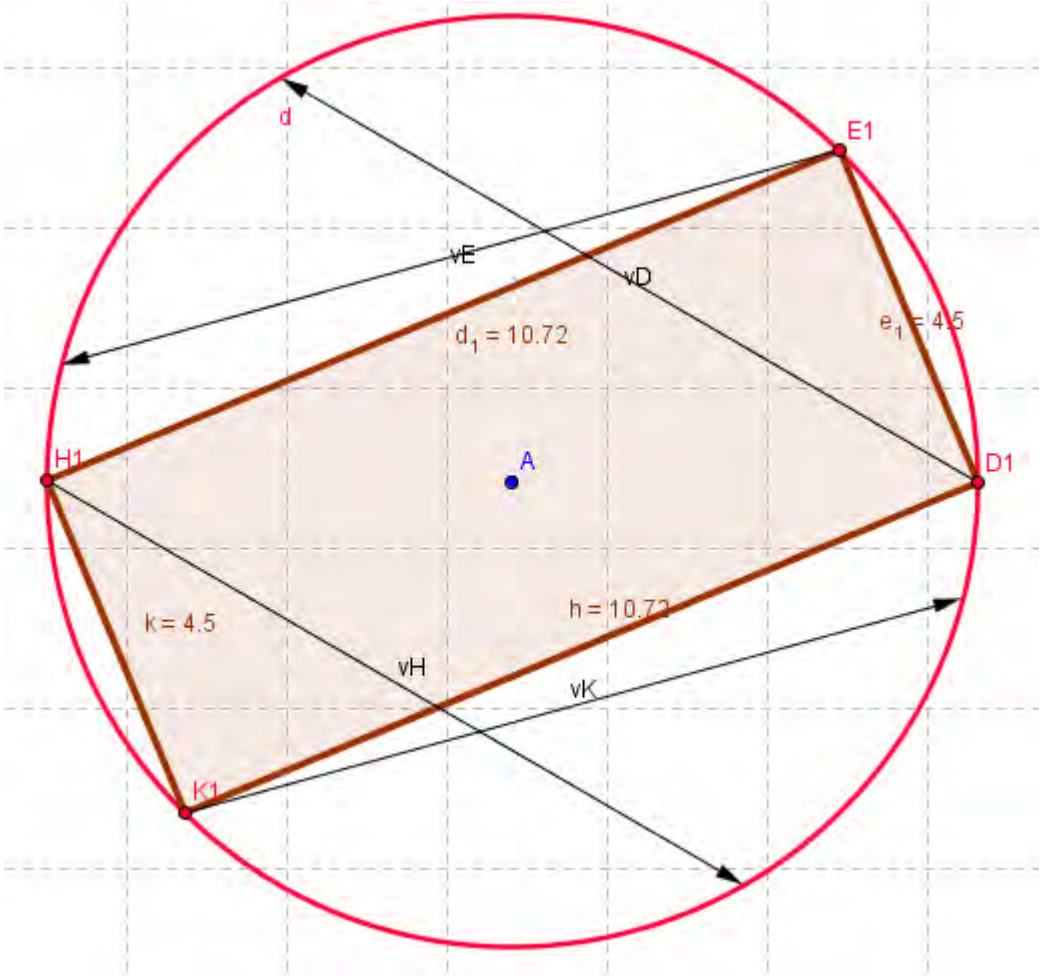
Rotación

destino Q=[3.91 1.17 9.69 12.43 ;13.88 10.3 3.8 7.38]



Rotación

vD	10.0645	149.8173	-8.7000	5.0600	12.6100	8.8200
vE	10.0800	195.3598	-9.7200	-2.6700	10.8900	12.9700
vH	10.0645	329.8173	8.7000	-5.0600	0.9900	8.8600
vK	10.0800	15.3598	9.7200	2.6700	2.7100	4.7100



- vD = (10.06; 149.82°)
- vE = (10.08; 195.36°)
- vH = (10.06; 329.82°)
- vK = (10.06; 15.39°)

- vD = (-8.7, 5.06)
- vE = (-9.72, -2.67)
- vH = (8.7, -5.06)
- vK = (9.7, 2.67)

Rotación

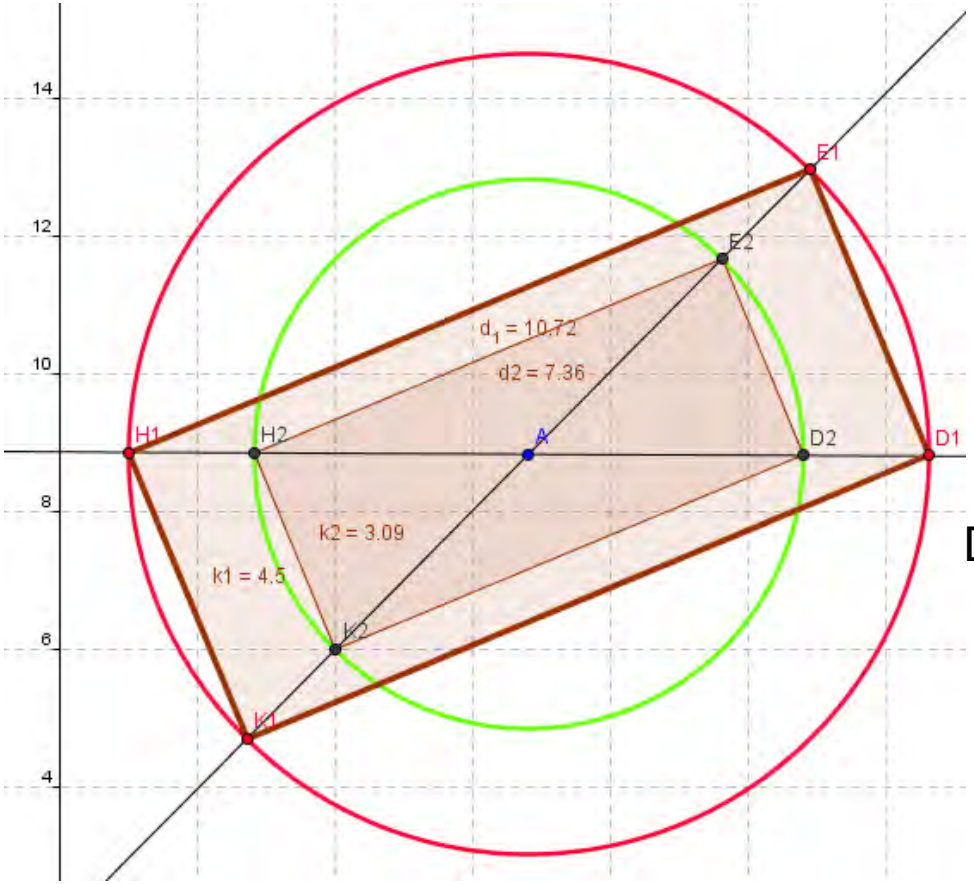
distancias =La diagonal en rojo son las distancias euclideas de los vectores que producen la rotación.

	(vD)D1	(vE) E1	(vH) H1	(vK) K1
(vD) D	10.0645	7.0391	5.8075	9.2482
(vE) E	11.5353	10.0800	1.4512	5.7982
(vH) H	5.8075	9.2482	10.0645	7.0391
(vK) K	1.4512	5.7982	11.5353	10.0800

Escala

origen=[12.61 10.89 0.99 2.71 ;8.82 12.97 8.86 4.71];

extremo=[10.79 9.61 2.81   3.99 ; 8.82 11.67 8.86 6.01];



Distancias del escalamiento

1.8200	4.1512	9.8001	9.0652
4.1379	1.8244	9.0664	9.8006
9.8001	9.0652	1.8200	4.1512
9.0664	9.8006	4.1379	1.8244

1.8200	179.9969	-1.8200	0.0001	12.6100	8.8200
1.8244	225.4441	-1.2800	-1.3000	10.8900	12.9700
1.8200	0.0031	1.8200	0.0001	0.9900	8.8600
1.8244	45.4441	1.2800	1.3000	2.7100	4.7100

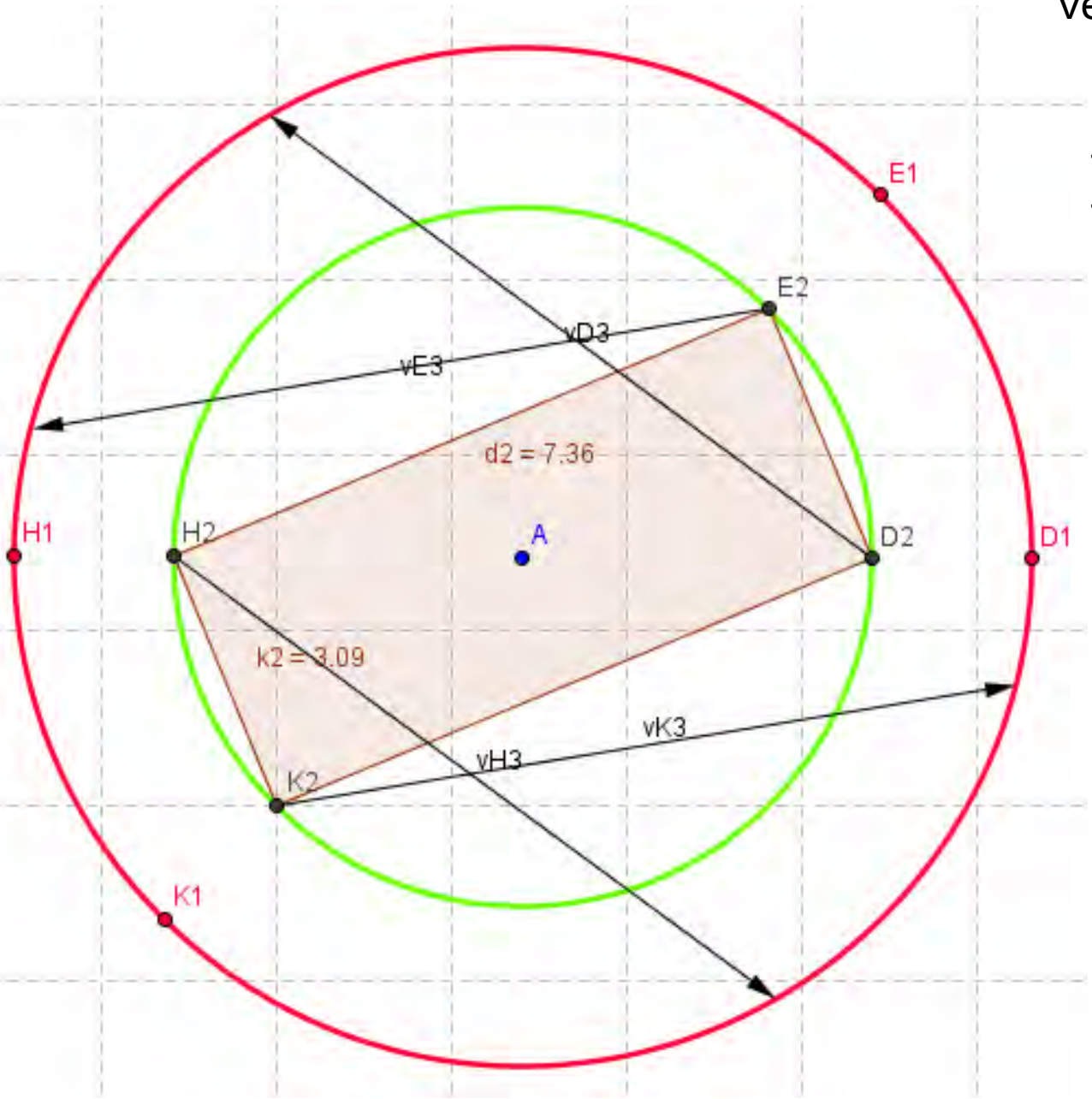
Rectángulo pequeño a rectángulo grande rotado:

origen=[10.79 9.61 2.81 3.99 ; 8.82 11.67 8.86 6.01];
extremo=[3.91 1.17 9.69 12.43 ;13.88 10.3 3.8 7.38];

8.5404	143.6668	-6.8800	5.0600	10.7900	8.8200
8.5505	189.2200	-8.4400	-1.3700	9.6100	11.6700
8.5404	323.6668	6.8800	-5.0600	2.8100	8.8600
8.5505	9.2200	8.4400	1.3700	3.9900	6.0100

distancias de escalamiento y rotación:

8.5404	6.1134	5.1391	7.8704
9.7332	8.5505	2.1825	5.1339
5.1391	7.8704	8.5404	6.1134
2.1825	5.1339	9.7332	8.5505



Vector de escalamiento V1

1.8200	0
1.2800	1.3000
-1.8200	0
-1.2800	-1.3000

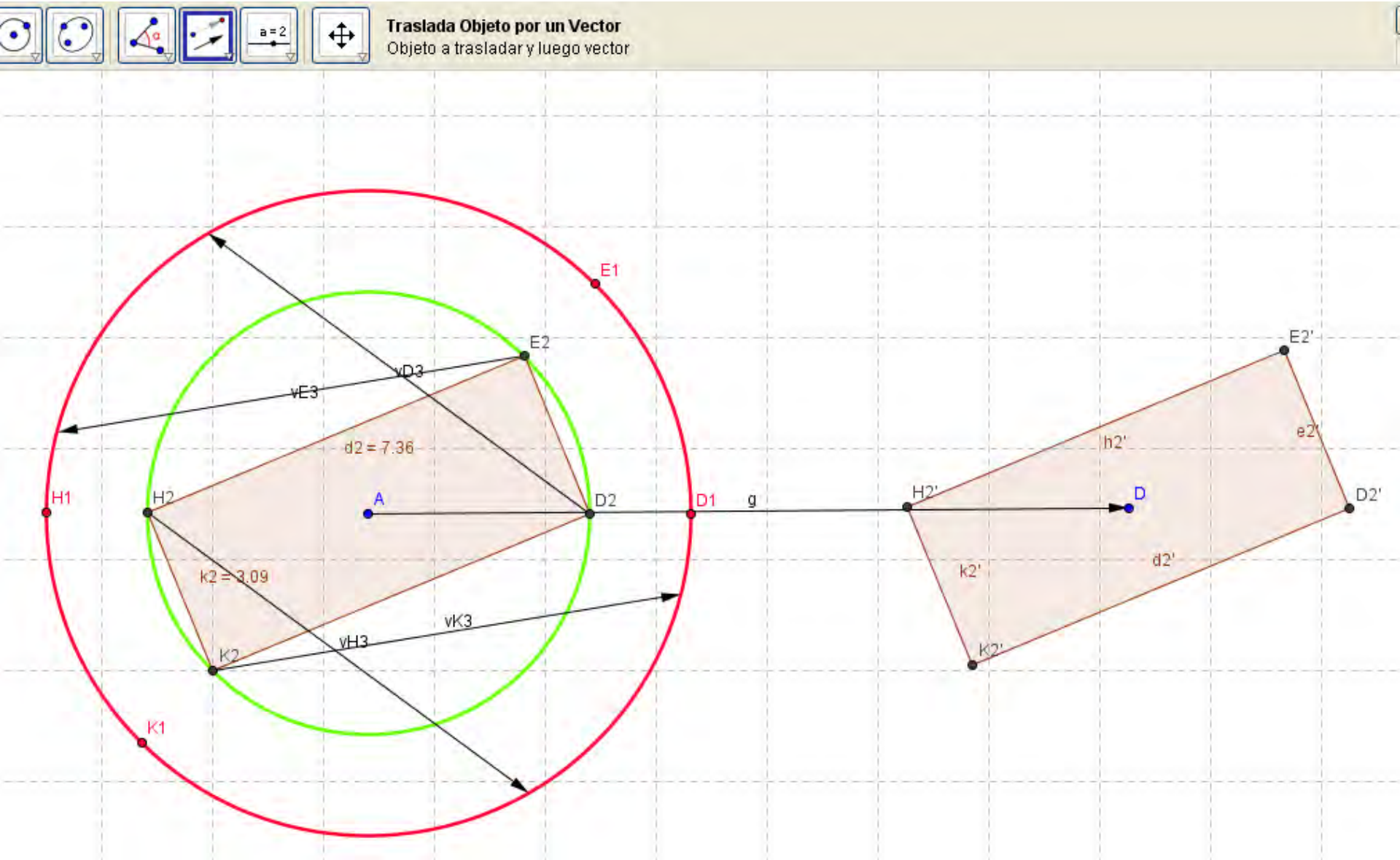
Vector de rotación V2

-8.7000	5.0600
-9.7200	-2.6700
8.7000	-5.0600
9.7200	2.6700

Vector final V3=V1+V2

-6.8800	5.0600
-8.4400	-1.3700
6.8800	-5.0600
8.4400	1.3700

Traslación


Traslada Objeto por un Vector
Objeto a trasladar y luego vector
a = 2
E1
E2
vD3
vE3
d2 = 7.36
H1
H2
A
D2
D1
g
H2'
D
E2'
h2'
e2'
D2'
k2 = 3.09
vK3
vH3
K2
k2'
d2'
K2'
K1

Translación

13.7004	180.4182	-13.7000	-0.1000	24.4900	8.9200
13.7004	180.4182	-13.7000	-0.1000	23.3100	11.7700
13.7104	180.4179	-13.7100	-0.1000	16.5200	8.9600
13.7104	180.4179	-13.7100	-0.1000	17.7000	6.1100

distancias =

13.7004	12.8628	5.7317	7.4224
15.1320	13.7004	7.4224	9.8164
21.6801	20.7055	13.7104	15.1418
20.7055	20.1604	12.8726	13.7104

Vector de traslación

-13.7000	-0.1000
-13.7000	-0.1000
-13.7100	-0.1000
-13.7100	-0.1000

```
%traslación del cuadrado pequeño
clear;origen=[24.49  23.31   16.52  17.7 ;8.92  11.77   8.96  6.11]; %origen
extremo=[ 10.79  9.61  2.81   3.99 ; 8.82  11.67  8.86  6.01];
%cuadrado pequeño como origen y cuadrado escalado grande como destino V1
clear; origen=[ 10.79  9.61  2.81   3.99 ; 8.82  11.67  8.86  6.01];
extremo=[12.61  10.89 0.99  2.71 ;8.82  12.97  8.86  4.71];
%cuadrado grande a cuadrado grande rotado V2
clear; origen=[12.61  10.89 0.99  2.71 ;8.82  12.97  8.86  4.71];
extremo=[3.91  1.17  9.69  12.43 ;13.88 10.3  3.8 7.38];%destino
```

Traslación (T), Escalado (E) y Rotación (R)

21.1693	166.4495	-20.5800	4.9600	24.4900	8.9200
22.1887	183.7986	-22.1400	-1.4700	23.3100	11.7700
8.5601	217.0707	-6.8300	-5.1600	16.5200	8.9600
5.4209	166.4508	-5.2700	1.2700	17.7000	6.1100

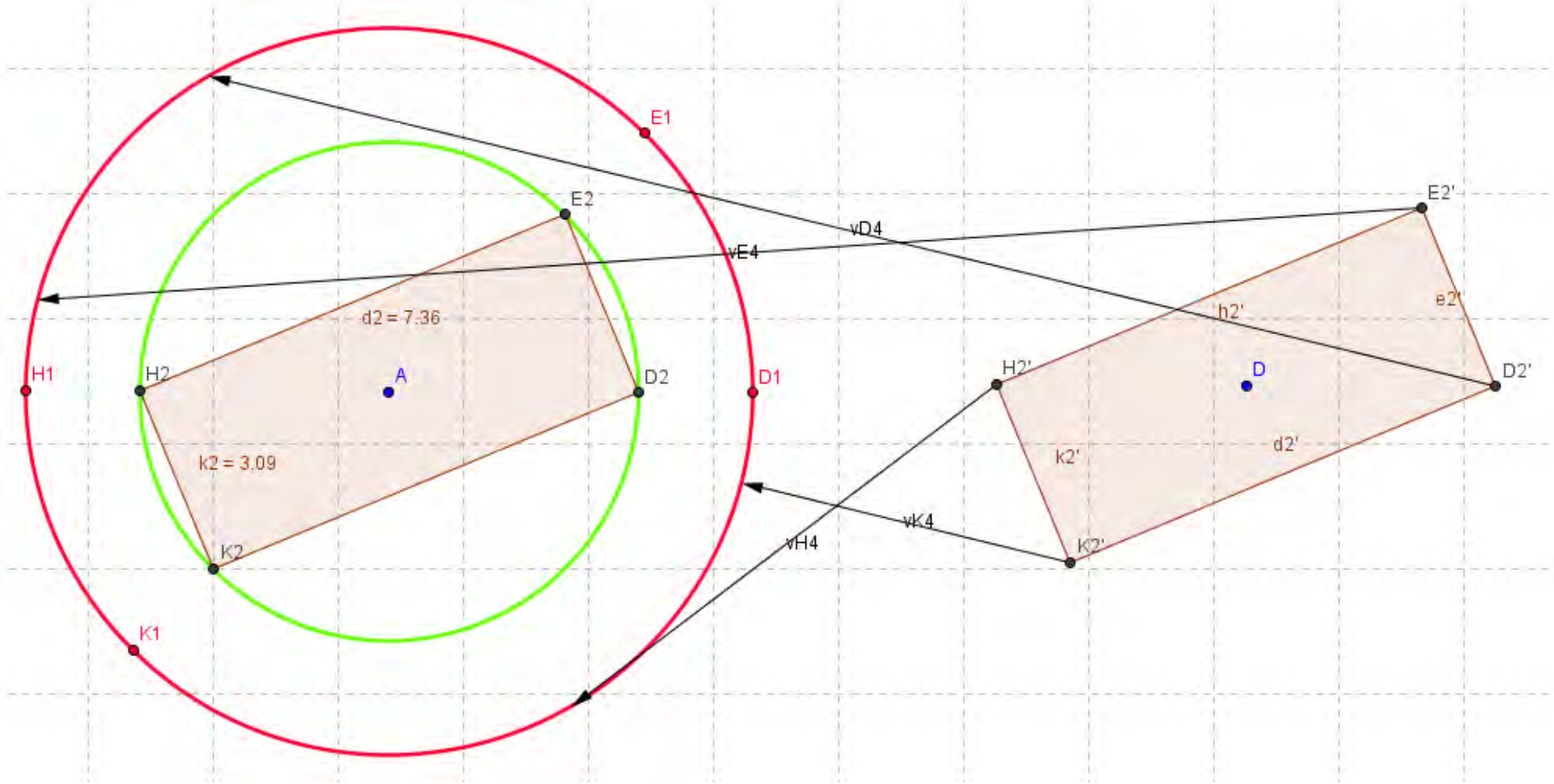
distancias (TER)

21.1693	19.5144	13.5358	15.8284
23.3608	22.1887	15.4084	17.0528
15.6606	15.7805	8.5601	8.3364
12.1579	11.7323	4.3846	5.4209

```
>> escala=(max(distancias)+min(distancias))./(max(distancias)-min(distancias))
   3.1705    3.2440    1.7955    1.9321
```

Vectores de traslación			Vectores de escalamiento			Vectores de rotación			Vectores suma	
-13.7000	-0.1000		1.8200	0		-8.7000	5.0600		-20.5800	4.9600
-13.7000	-0.1000	+	1.2800	1.3000	+	-9.7200	-2.6700	=	-22.1400	-1.4700
-13.7100	-0.1000		-1.8200	0		8.7000	-5.0600		-6.8300	-5.1600
-13.7100	-0.1000		-1.2800	-1.3000		9.7200	2.6700		-5.2700	1.2700

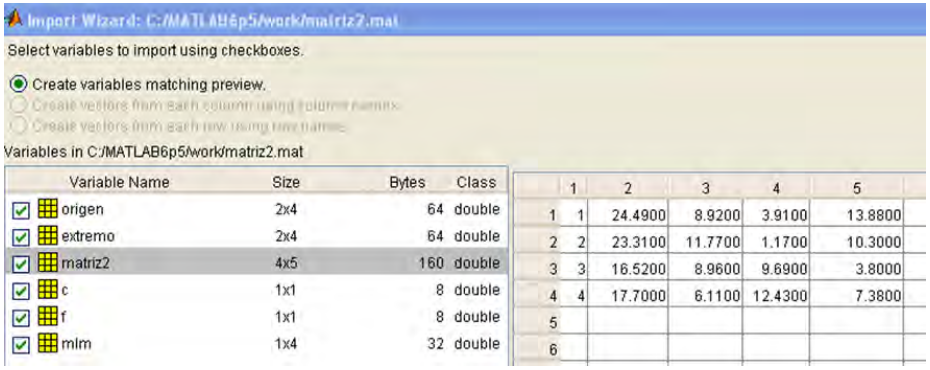
Traslación, escalado y rotación

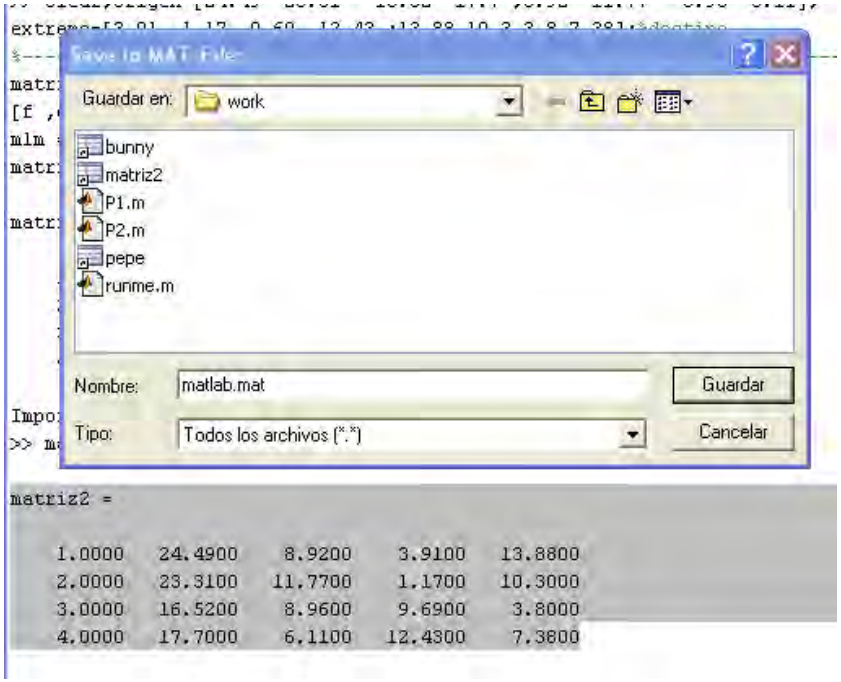


En matemática, el análisis procrusteano o «**procrustes**» (PS) es el nombre que se da al proceso de aplicar una transformación euclideana que conserva la forma eliminando las diferencias de escala, rotación y traslación.

```
>> clear;origen=[24.49  23.31   16.52  17.7 ;8.92  11.77   8.96  6.11];
extremo=[3.91  1.17  9.69  12.43 ;13.88 10.3 3.8 7.38];%destino
%----------formamos la   Matriz2-----------------------------------------------
matriz2=[origen' extremo'];
[f ,c ] =size(matriz2);
mlm = 1:1:f ;  %vector de numeros para cada landmark
matriz2=[mlm' matriz2]
matriz2 =  % tipo de ordenación con la que vamos a trabajar, sería la entrada de datos.
%nº de          imagen1              imagen2          .............etc
%landMark     x          y          x          y
    1.0000   24.4900    8.9200    3.9100   13.8800
    2.0000   23.3100   11.7700    1.1700   10.3000
    3.0000   16.5200    8.9600    9.6900    3.8000
    4.0000   17.7000    6.1100   12.4300    7.3800
```

Guardamos la matriz (como *.mat) y
La podemos importar de nuevo





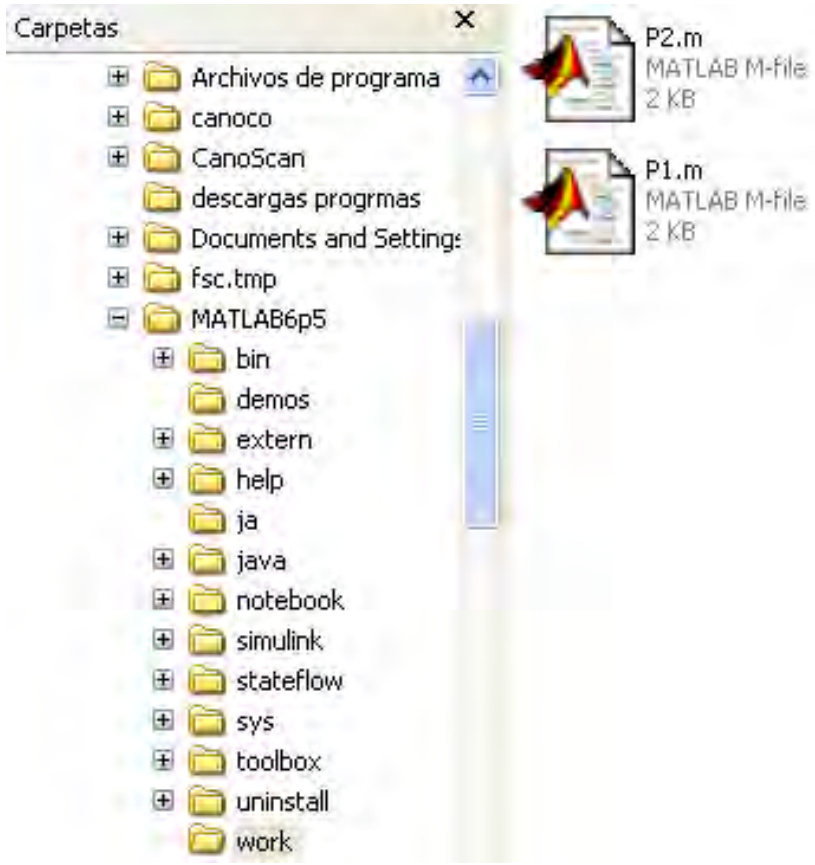
Funciones:

```
>> total=P1(origen,extremo)  %función P1
   21.1693  166.4495  -20.5800    4.9600   24.4900    8.9200
   22.1887  183.7986  -22.1400   -1.4700   23.3100   11.7700
    8.5601  217.0707   -6.8300   -5.1600   16.5200    8.9600
    5.4209  166.4508   -5.2700    1.2700   17.7000    6.1100
```

```
function total =  P1 (origen,extremo);
%código de la matriz_4.tx
return;
```

```
>> distancias=P2(origen,extremo)%función P2
   21.1693   19.5144   13.5358   15.8284
   23.3608   22.1887   15.4084   17.0528
   15.6606   15.7805    8.5601    8.3364
   12.1579   11.7323    4.3846    5.4209
```

```
function distancias =  P2 (origen,extremo);
% código de la matriz_4.tx
return;
```



Ver funciones una vez copiadas a
La carpeta work de matlab.

>> det(TER)
-19.6399

Distancias mínimas en el conjunto del movimiento

>> TER

21.1693	19.5144	13.5358	15.8284
23.3608	22.1887	15.4084	17.0528
15.6606	15.7805	8.5601	8.3364
12.1579	11.7323	4.3846	5.4209

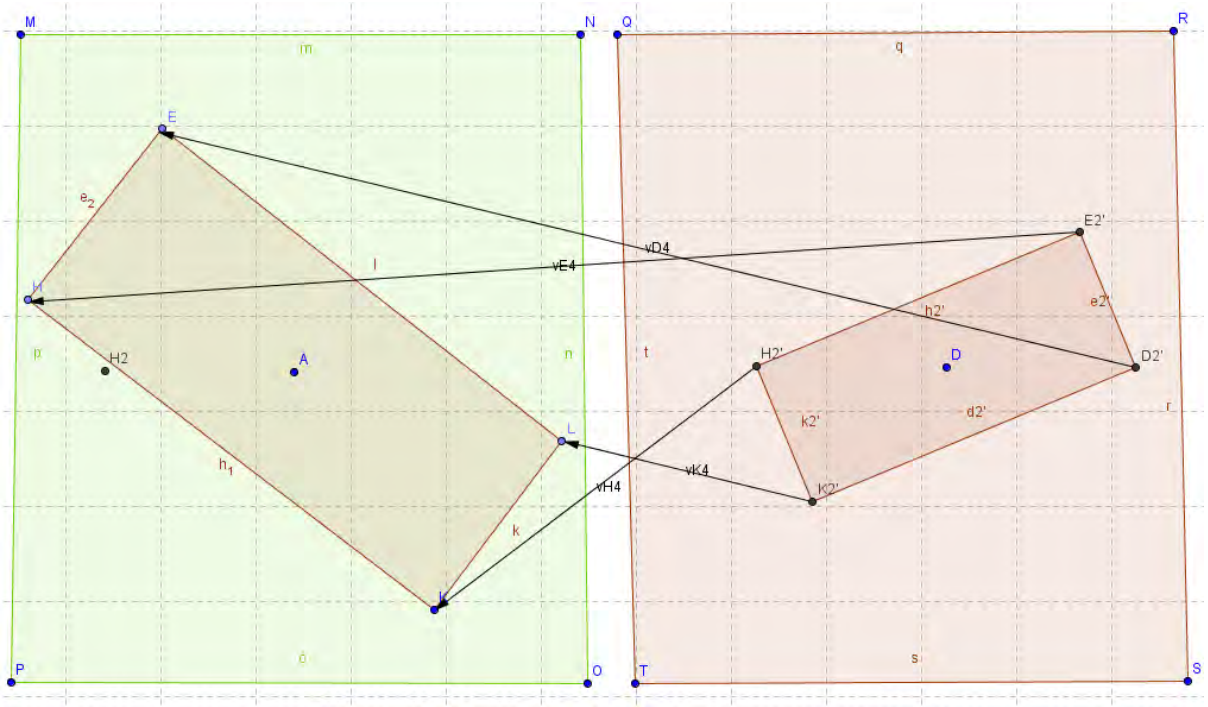
>> suma=T+E+R Distancias totales

>> suma/3

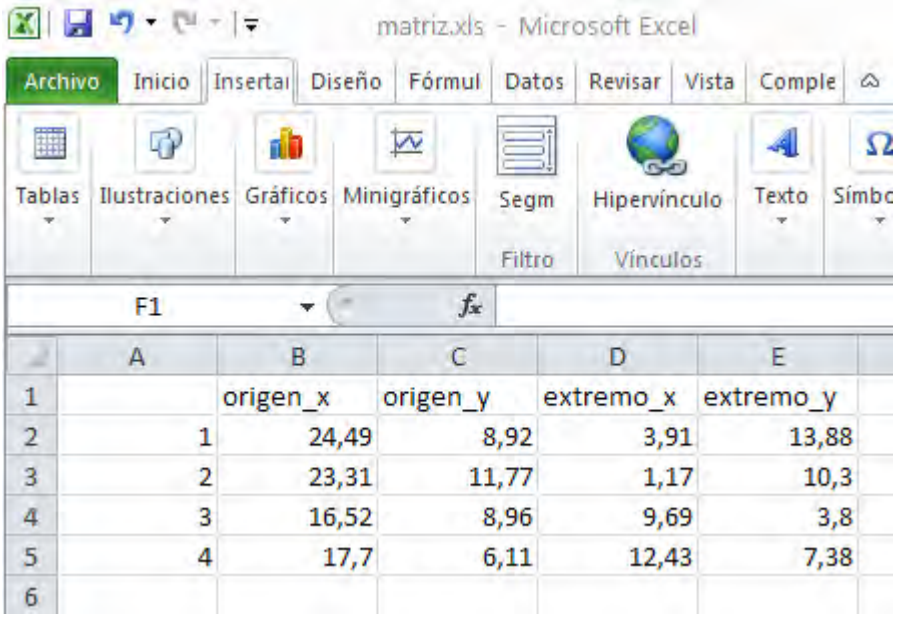
8.5283	8.0133	7.1131	8.5790
10.2728	8.5349	5.9796	8.4717
12.4292	13.0067	8.5316	8.7729
10.4073	11.9197	9.5197	8.5383

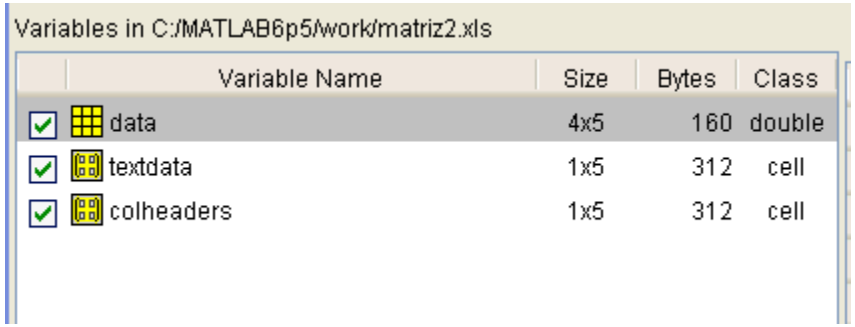
>> det(suma/3)
-24.0283
>> det(suma/3)/det(TER)
1.2234
>> det(suma/3)*det(inv(TER))
1.22

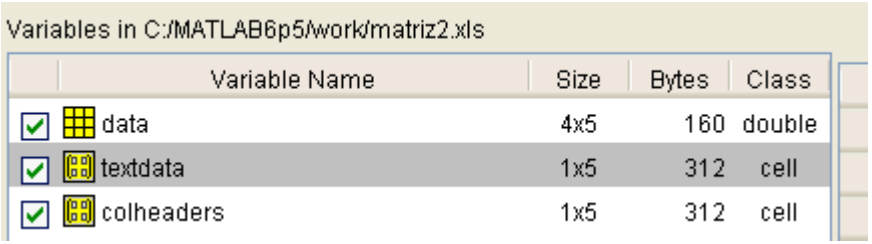
Indice vectorial que indica la disimilaridad de dos geometrías iguales cuando dentro de ellas existen dos geometrías semejantes.



Podemos importar la matriz desde un archivo *.xls








```
>> textdata
    [ ]    'origen_x'    'origen_y'    'extremo_x'    'extremo_y'
>> data
    1.0000   24.4900    8.9200    3.9100   13.8800
    2.0000   23.3100   11.7700    1.1700   10.3000
    3.0000   16.5200    8.9600    9.6900    3.8000
    4.0000   17.7000    6.1100   12.4300    7.3800
```

Tambien podemos crear una matriz de texto

```
>> c = {'landmarks','origen x','origen y','extremo x','extremo y'; }
    'landmarks'    'origen x'    'origen y'    'extremo x'    'extremo y'

>> c(2)
    'origen x'
>> textdata(2)
    'origen_x'
```

Y darle una presentación diferente

```
>> fields = {'nombre'};
>> s = cell2struct(c, fields, 5);
>> s(1)
    nombre: 'landmarks'
>> s(2)
    nombre: 'origen x'
>> s(3)
    nombre: 'origen y'
>>  etc..
```

%Traslación del 1º landmark a origen xy

```
>>fprintf(1, ' definimos submatrices x e y')
>> [f ,c ] =size(matriz2);
>> x= matriz2(:,2:2:c)          >> y= matriz2(:,3:2:c)
   24.4900    3.9100               8.9200   13.8800
   23.3100    1.1700              11.7700   10.3000
   16.5200    9.6900               8.9600    3.8000
   17.7000   12.430                6.1100    7.3800

%Tomamos el primer landmark y lo pasamos al origen xy
>> [fxy,cxy]=size(x);
lm1= zeros(fxy,cxy);x1=x(1,1:1:cxy);
for i=1:fxy
lm1(i,:)=x1;
end;
x0=zeros(fxy,cxy);
x0(:,1:1:cxy)=x(:,1:1:cxy)-lm1(:,1:1:cxy)
         0         0
   -1.1800   -2.7400
   -7.9700    5.7800
   -6.7900    8.5200

>> y1=y(1,1:1:cxy);
for i=1:fxy
lm1(i,:)=y1;
end;
y0=zeros(fxy,cxy);
y0(:,1:1:cxy)=y(:,1:1:cxy)-lm1(:,1:1:cxy)
         0         0
    2.8500   -3.5800
    0.0400  -10.0800
   -2.8100   -6.5000
```

%Rotamos el 2º landmark hasta el eje de las x, vemos las distancias entre puntos ,luego el tercero, %etc... hasta que las distancias sean mínimas tomando las matrices x0 e y0. **La funcion P3 retorna matriz %polares. P1 (retorna matriz2 sin landmarks,P2 y P3 tienen el mismo algoritmo. P2 retorna distancias %(euclideas).** %Definimos la matriz origen y la matriz extremo:

```
>> origen=[x0(1:1:fxy,1)  y0(1:1:fxy,1)]'
         0   -1.1800   -7.9700   -6.7900
         0    2.8500    0.0400   -2.8100
>> extremo=[x0(1:1:fxy,2)  y0(1:1:fxy,2)]'
         0   -2.7400    5.7800    8.5200
         0   -3.5800  -10.0800   -6.5000

>> polares=P3(origen,extremo)%función P3
         0         0
    6.6165  256.3628
   17.0727  323.6469
   15.7484  346.4491
>> sumdis =sum(polares)
   39.4376  926.4588
```

```
>> sumdis0=sumdis(1,1)
   39.4376  distancias entre puntos sin rotar y el primer landmark a 0 en ambas figuras
>>origen0=origen;extremo0=extremo;
```

%rotación 1 de la primera figura

```
>> origen=zeros(cxy,fxy)
     0     0     0     0
     0     0     0     0
>> extremo=origen0 %pasa a ver vectores de
%los  landmarks de la figura 1 inicial
         0   -1.1800   -7.9700   -6.7900
         0    2.8500    0.0400   -2.8100

>> polares=P3(origen,extremo)%función P3;
>> polares0=polares
         0         0
    3.0846  112.4913
    7.9701  179.7124
    7.3485  202.4819

    >> r1=zeros(fxy,cxy)
    for i=2:fxy
    r1(i,2)=polares0(2,2);
    end;
    polares1=polares0-r1
              0         0
       3.0846         0
       7.9701   67.2212
       7.3485   89.9906
```

Cuando se conoce el módulo del vector y el ángulo α que forma con el eje OX, las coordenadas de P son:
x = módulo · cos α
y = módulo · sen α
En radianes grados*2pi/360

```
>> xy1=zeros(fxy,cxy);
rpolares1=polares1;
for i=2:fxy
rpolares1(i,2)=((polares1(i,2))*2*pi)/360;
end;
xy1=zeros(fxy,cxy);
for i=2:fxy;
xy1(i,1)=((polares1(i,1))*(cos(rpolares1(i,2))));
xy1(i,2)=((polares1(i,1))*(sin(rpolares1(i,2))));
end;
>> xy1  %matriz fig1 rotada
         0         0
    3.0846         0
    3.0858    7.3485
    0.0012    7.3485

>> origen1=xy1'
         0    3.0846    3.0858    0.0012
         0         0    7.3485    7.3485
```

%rotación 1 de la segunda figura

```
>> origen=zeros(cxy,fxy);
     0     0     0     0
     0     0     0     0
>> extremo=extremo0; %pasa a ver vectores de
%los  landmarks de la figura 2 inicial
>> polares=P3(origen,extremo)%función P3
         0         0
    4.5082   232.5709 % rotación 1
   11.6196   299.8305 %rotación 2
   10.7164   322.6596  %rotación 3
>>polares0=polares;

  >> r1=zeros(fxy,cxy);
  >>  for i=2:fxy
  r1(i,2)=polares(2,2);
  end;
  >> r1
          0         0
          0  232.5709
          0  232.5709
          0  232.5709

   >> polares1 =polares0-r1
         0            0
      4.5082         0
     11.6196   67.2596
     10.7164   90.0887
```

```
>> for i=2:fxy
rpolares1(i,2)=((polares1(i,2))*2*pi)/360;
end
>> rpolares1
         0            0
    4.5082         0
   11.6196    1.1739
   10.7164    1.5723

>> xy1=zeros(fxy,cxy);
for i=2:fxy;
xy1(i,1)=((polares1(i,1))*(cos(rpolares1(i,2))));
xy1(i,2)=((polares1(i,1))*(sin(rpolares1(i,2))));
end;
>> xy1
           0         0
     4.5082         0
     4.4916   10.7163
    -0.0166   10.7163

>> extremo1=xy1'
         0     4.5082    4.4916   -0.0166
         0         0    10.7163   10.7163
```

COMPARAMOS IMAGEN 1 ROTADA CON LA 2 ROTADA

```
>> extremo=extremo1;% IMAGEN 2
origen=origen1; % IMAGEN 1
polares=P3(origen,extremo)%función P3;
sumdis =sum(polares);
sumdis1=sumdis(1,1)
    8.4410 %las  distancias entre landmarks se reducen
```

REPETIMOS CON EL LANDMARK 3 LA ROTACIÓN DE LA 2ª IMAGEN

```
>> origen=zeros(cxy,fxy);
extremo=extremo0;
polares=P3(origen,extremo)%función P3
polares0=polares;
r1=zeros(fxy,cxy)
for i=2:fxy
r1(i,2)=polares0(3,2);
end;
         0         0
    4.5082  232.5709
   11.6196  299.8305
   10.7164  322.6596
polares1=polares0-r1
         0         0
    4.5082  -67.2596  %el ángulo no puede
   11.6196         0       %ser negativo
   10.7164   22.8291
```

```
>> for i=1:fxy
 if polares1(i,2)<0
    polares1(i,2)=360+polares1(i,2)
 end;
end;
polares1 =
         0         0
    4.5082  292.7404  %se corrigió
   11.6196         0
   10.7164   22.8291
>> xy1=zeros(fxy,cxy);
rpolares1=polares1;
for i=2:fxy
rpolares1(i,2)=((polares1(i,2))*2*pi)/360;
end;
xy1=zeros(fxy,cxy);
for i=2:fxy;
xy1(i,1)=((polares1(i,1))*(cos(rpolares1(i,2))));
xy1(i,2)=((polares1(i,1))*(sin(rpolares1(i,2))));
end;
extremo2=xy1'
         0    1.7427   11.6196    9.8769
         0   -4.1578         0    4.1578
```

PARA ABREVIAR CODIGOS Y PASOS TENEMOS QUE RESUMIR TODO.

```
>> matriz2   %importamos la matriz2 que nos vale de punto de partida
    1.0000   24.4900    8.9200    3.9100   13.8800
    2.0000   23.3100   11.7700    1.1700   10.3000
    3.0000   16.5200    8.9600    9.6900    3.8000
    4.0000   17.7000    6.1100   12.4300    7.3800
```

```
>> [f ,c ] =size(matriz2)  %vemos las dimensiones de la matriz2
```

f = 4 c = 5

```
>> x= matriz2(:,2:2:c);
y= matriz2(:,3:2:c);%definimos matrices x e y
```

x =

```
   24.4900    3.9100
   23.3100    1.1700
   16.5200    9.6900
   17.7000   12.4300
```

y =

```
    8.9200   13.8800
   11.7700   10.3000
    8.9600    3.8000
    6.1100    7.3800
```

```
>>[fxy,cxy]=size(x);  %vemos dimensiones de x e y
```

fxy = 4 cxy = 2

```
>> x1=x(1,1:1:cxy); %definimos las x del primer landmark
```

x1 = 24.4900 3.9100

```
>> for i=1:fxy
lm1(i,:)=x1;   %definimos una matriz con los valores repetidos
end;
```

lm1 =

```
   24.4900    3.9100
   24.4900    3.9100
   24.4900    3.9100
   24.4900    3.9100
```

```
>> x0(:,1:1:cxy)=x(:,1:1:cxy)-lm1(:,1:1:cxy); % y restamos
```

Hemos trasladado al punto origen las x

x0 =

```
         0         0
   -1.1800   -2.7400
   -7.9700    5.7800
   -6.7900    8.5200
```

```
>> y1=y(1,1:1:cxy); %hacemos lo mismo con las y
for i=1:fxy
lm1(i,:)=y1;
end;
y0(:,1:1:cxy)=y(:,1:1:cxy)-lm1(:,1:1:cxy);
```

```
y0 =
         0         0
    2.8500   -3.5800
    0.0400  -10.0800
   -2.8100   -6.5000
```

Hemos trasladado al punto origen las y: tenemos ya un x0 y un y0 con dos nuevas matrices , la primera que le denominamos origen0 y corresponde a la figura 1 y la segunda que le denominamos extremo0 y pertenece a la figura 2

```
>>  origen0=[x0(1:1:fxy,1)  y0(1:1:fxy,1)]';
extremo0=[x0(1:1:fxy,2)  y0(1:1:fxy,2)]';
```

Hemos terminado la traslación y empezamos las rotaciónes a partir del segundo landmark (fxy-1 rotaciones)

```
origen0 =          las x  encima y las y debajo
         0   -1.1800   -7.9700   -6.7900
         0    2.8500    0.0400   -2.8100
extremo0 =
         0   -2.7400    5.7800    8.5200
         0   -3.5800  -10.0800   -6.5000
```

```
>> matrot= [origen0'  extremo0'  zeros(fxy,4*(fxy-1))]' ;  matriz de rotaciones
>>lm=5;
for rot=2:1:fxy
   for r=1:2:4
       origen=zeros(cxy,fxy);   extremo=matrot(r:1:(r+1),1:fxy);
       polares=P3(origen,extremo);
       for i=2:fxy
         r1(i,2)=polares(rot,2);
       end;
       polares=polares-r1;
       for i=1:fxy
         if polares(i,2)<0
         polares(i,2)=360+polares(i,2);  end;
       end;
```

```
matrot =
         0   -1.1800   -7.9700   -6.7900
         0    2.8500    0.0400   -2.8100
         0   -2.7400    5.7800    8.5200
         0   -3.5800  -10.0800   -6.5000
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
```

```
                for i=2:fxy
                  rpolares(i,2)=((polares(i,2))*2*pi)/360;
                end;
                for i=2:fxy;
                  xy1(i,1)=((polares(i,1))*(cos(rpolares(i,2))));
                  xy1(i,2)=((polares(i,1))*(sin(rpolares(i,2))));
                end;
        matrot(lm:lm+1,1:fxy)=xy1';matrot1(lm:lm+1,1:fxy)=r1';%matrices
        lm=lm+2;
   end;
end;

fila=1;
sumdis2=zeros(fxy,1);
for r=1:4:(4*fxy)-1 % para procesar matrot y ver distancias
                origen=matrot(r:r+1,1:fxy);
                extremo=matrot(r+2:r+3,1:fxy);
                polares=P3(origen,extremo);
                sumdis =sum(polares);sumdis1=sumdis(1,1);
                sumdis2(fila,1)=sumdis1
                fila=fila+1
end;
```

```
 >> minimo=min(sumdis2);p=0;
 for i=1:fxy
    if minimo==sumdis2(i,1)
     break;end;
 p=p+1;
 end;
```

sumdis2 = suma de distancias

```
   39.4376
    8.4410
    8.4410
    8.4410
```

p = 1ª distancia mínima

```
    2
```

matrot = matriz de rotaciones

```
         0   -1.1800   -7.9700   -6.7900
         0    2.8500    0.0400   -2.8100
         0   -2.7400    5.7800    8.5200
         0   -3.5800  -10.0800   -6.5000
         0    3.0846    3.0858    0.0012
         0         0    7.3485    7.3485
         0    4.5082    4.4916   -0.0166
         0         0   10.7163   10.7163
         0    1.1943    7.9701    6.7758
         0   -2.8440         0    2.8440
         0    1.7427   11.6196    9.8769
         0   -4.1578         0    4.1578
         0    0.0005    7.3490    7.3485
         0   -3.0846   -3.0846         0
         0   -0.0070   10.7094   10.7164
         0   -4.5082   -4.5082         0
```

matrot1 = ángulos de rotación

```
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0  112.4913  112.4913  112.4913
         0         0         0         0
         0  232.5709  232.5709  232.5709
         0         0         0         0
         0  179.7124  179.7124  179.7124
         0         0         0         0
         0  299.8305  299.8305  299.8305
         0         0         0         0
         0  202.4819  202.4819  202.4819
         0         0         0         0
         0  322.6596  322.6596  322.6596
```

```
>> f=(2*p)+1;
matmin=matrot(f:f+3,1:fxy); matmin1=matrot1(f+1:2:f+3,1:fxy);
```

AHORA HAY QUE ESCALAR LA IMAGEN 1

```
>>origen1=matmin(1:2,1:fxy);  extremo1=matmin(3:4,1:fxy);
>>origen=zeros(2,fxy);extremo=origen1;
>>distancias1=P2(origen,extremo);
>>extremo=extremo1;
>>distancias2=P2(origen,extremo);
>>max1=max(distancias1);max2=max(distancias2);
>>escala=max2/max1;
>>distancias1*escala
>>origen1*escala
>>extremo1
```

matmin = matrizes de minimas distancias entre los landmarks de la imagen 1 y la 2

```
     0    3.0846    3.0858    0.0012
     0         0    7.3485    7.3485
     0    4.5082    4.4916   -0.0166
     0         0   10.7163   10.7163
```

matmin1 = ángulos de rotación utilizados

```
     0  112.4913  112.4913  112.4913
     0  232.5709  232.5709  232.5709
```

distancias1 =

```
         0         0         0         0
    3.0846    3.0846    3.0846    3.0846
    7.9701    7.9701    7.9701    7.9701
    7.3485    7.3485    7.3485    7.3485
```

distancias2 =

```
         0         0         0         0
    4.5082    4.5082    4.5082    4.5082
   11.6196   11.6196   11.6196   11.6196
   10.7164   10.7164   10.7164   10.7164
```

max1 =

```
    7.9701    7.9701    7.9701    7.9701
```

max2 =

```
   11.6196   11.6196   11.6196   11.6196
```

max2/max1= escala

```
    1.4579
```

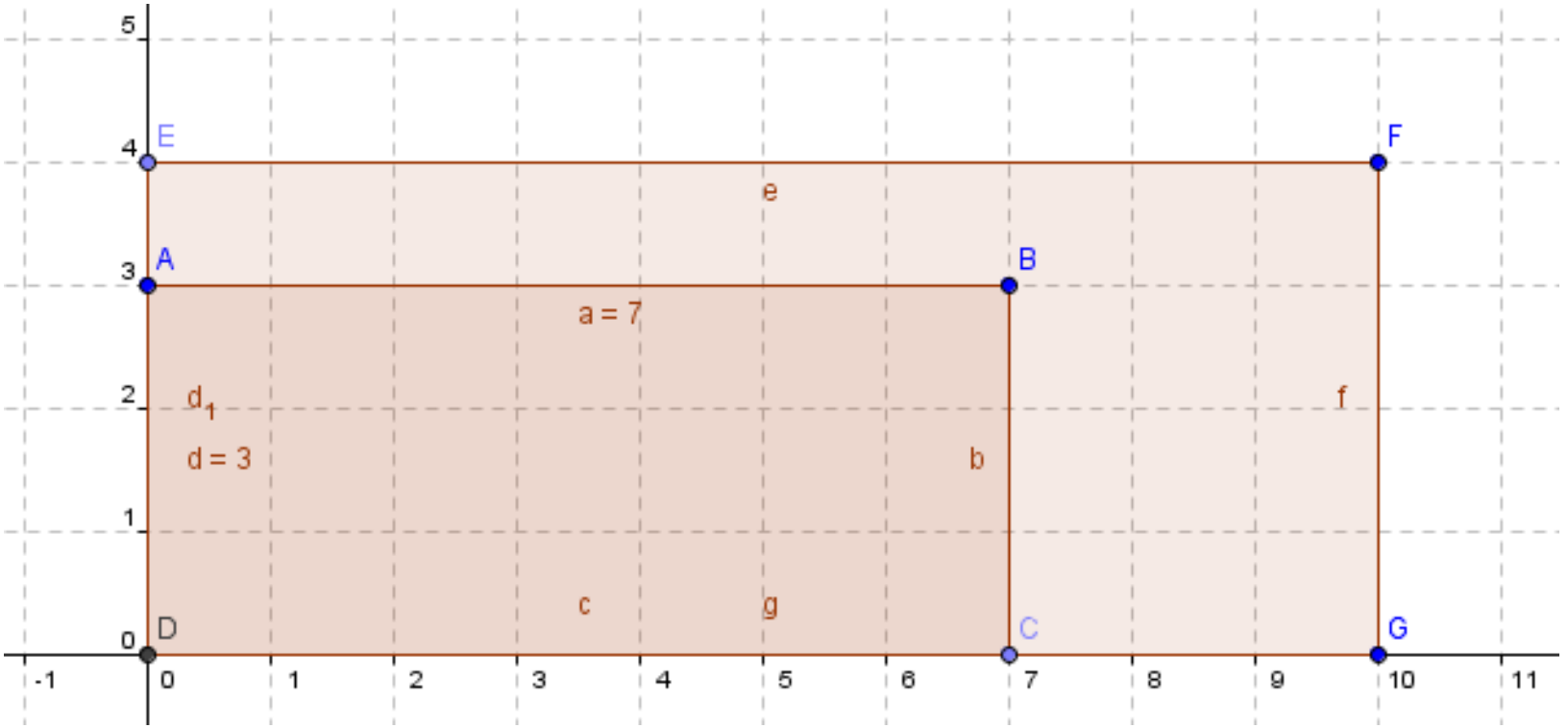
origen1*escala ~ extremo1

```
    0    4.4971    4.4988    0.0017          0    4.5082    4.4916   -0.0166
    0         0   10.7133   10.7133          0         0   10.7163   10.7163
```

```
>> origen=origen1*escala;
extremo=extremo1;
nuevasdistancias=P2(origen,extremo)
         0    4.4971   11.6196   10.7133
    4.5082    0.0112   10.7133   11.6226
   11.6196   10.7163    0.0078    4.4899
   10.7164   11.6281    4.5154    0.0186
```

diag(nuevasdistancias) =distancias entre puntos de ambas imágenes

```
         0
    0.0112
    0.0078
    0.0186
```




```
>> nlm=matriz2(1:fxy,1); nd=diag(nuevasdistancias);
>> matriz3=[nlm nd  origen' extremo' ]
    1.0000         0         0         0         0         0
    2.0000    0.0112    4.4971         0    4.5082         0
    3.0000    0.0078    4.4988   10.7133    4.4916   10.7163
    4.0000    0.0186    0.0017   10.7133   -0.0166   10.7163
```

Matriz devuelta por la función P4 procrustes de la matriz2

>> matriz=P4(matriz2)