Introducción a Matlab

2º de II e ITIS
Curso 06/07

Marta Penas Centeno

Matrices (I)

- Creación de matrices:
  - M1 = [1 2 3; 4 5 6]
  - M2 = 2 * ones(2,3); M3 = 1 + zeros(3,2)
  - v1 = 1:10; v2 = 0:0.1:0.9;
- Dimensiones de una matriz:
  - [filas columnas] = size(M1)
  - longitud = length(v1)

Matrices (II)

- Operaciones entre matrices:
  - M1 + M2; M1 – M2; M1 * M3;
  - M1 / M2 = M1 * inv(M2);
  - M1 \ M2 = inv(M1) * M2;
- Operaciones elemento a elemento:
  - M1 .* M2; M1 ./ M2; M1 .\ M2; M1 .^M2

Matrices (III)

- Operaciones matriz – escalar:
  - M1 .^ 2; M1 * 2
- Trasponer una matriz:
  - M4 = M1'
- Submatrices:
  - M7 = M1(1:2,2:3)
  - M8 = M1(2,1:3)
- Más información: help ops, help elmat

Lenguaje de programación (I)

- Sentencia IF:

```
IF expresion
   sentencias
ELSEIF expresion
   sentencias
ELSE
   sentencias
END
```

Lenguaje de programación (II)

- Sentencia FOR:

```
FOR variable = expresion,
   sentencias
END;
```

- Ejemplo:

```
FOR I = 1:0.5:N,
   FOR J = 1:N,
      sentencias
   END
END
```

Lenguaje de programación (IV)

- Sentencia WHILE:

  WHILE expresion,

  sentencias

  END

Lenguaje de programación (V)

- Sentencia SWITCH:

```
SWITCH switch_expr
   CASE case_expr,
      sentencias
   CASE {case_expr1, case_expr2,...}
      sentencias
   OTHERWISE
      sentencias
END
```

Entrada/Salida de datos (I)

- Llamada a funciones:
  - Fichero funcion.m en directorio actual
  - Prompt de matlab:

```
result = funcion(param)
```

Entrada/Salida de datos (II)

- Ejemplo:
  - media.m:

```
function md = media(x);
    n = length(x);
    md = sum(x)/n;
```

  - Matlab:

```
x = 10 * rand(1,10);
med = media(x);
```

Entrada/Salida de datos (III)

- Acceso a ficheros:
  - fopen: fid = fopen(filename,permission);
    - fid1 = fopen('input.txt','r');
    - fid2 = fopen('output.txt','w');
  - fclose: st = fclose(fid);
    - devuelve 0 si éxito y -1 si fallo.

Entrada/Salida de datos (IV)

- Acceso a ficheros:
  - fread: A = fread(fid[,size[,precision]])

    size: N, inf, [M,N].

    precision: int8,uint32,double, char, long...
  - fscanf: [A,count] = fscanf(fid,format,size)

    format: '%s', '%5d', '%f'.

    count: número de elementos que se han leído.

Entrada/Salida de datos (V)

- Acceso a ficheros:
  - fwrite: count = fwrite(fid,A,precision)
    - Los datos se escriben por columnas
  - fprintf: count = fprintf(fid, format,A)
    - \n, \t...

Entrada/salida de datos (VI)

- Imágenes:
  - Leer una imagen: IM = imread('fichero.ext');
    - Escala de grises: size(IM) => filas, columnas
    - Color: size(IM) => filas, columnas,3
    - Tipos de ficheros: pgm, ppm, gif, png...
  - Salvar una imagen: imwrite(IM,'fichero.ext');
  - Mostrar una imagen por pantalla: imshow(IM);

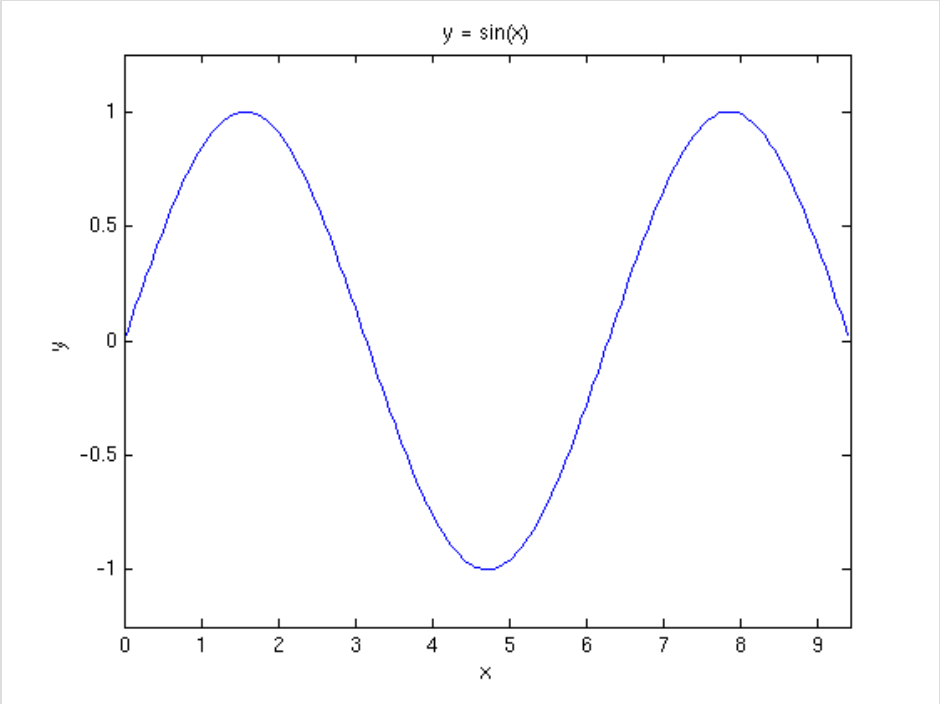
Gráficas (I)

- Funciones de una variable:

```
x = 0:0.05:3*pi;
y = sin(x);
plot(x,y,'b-');
```

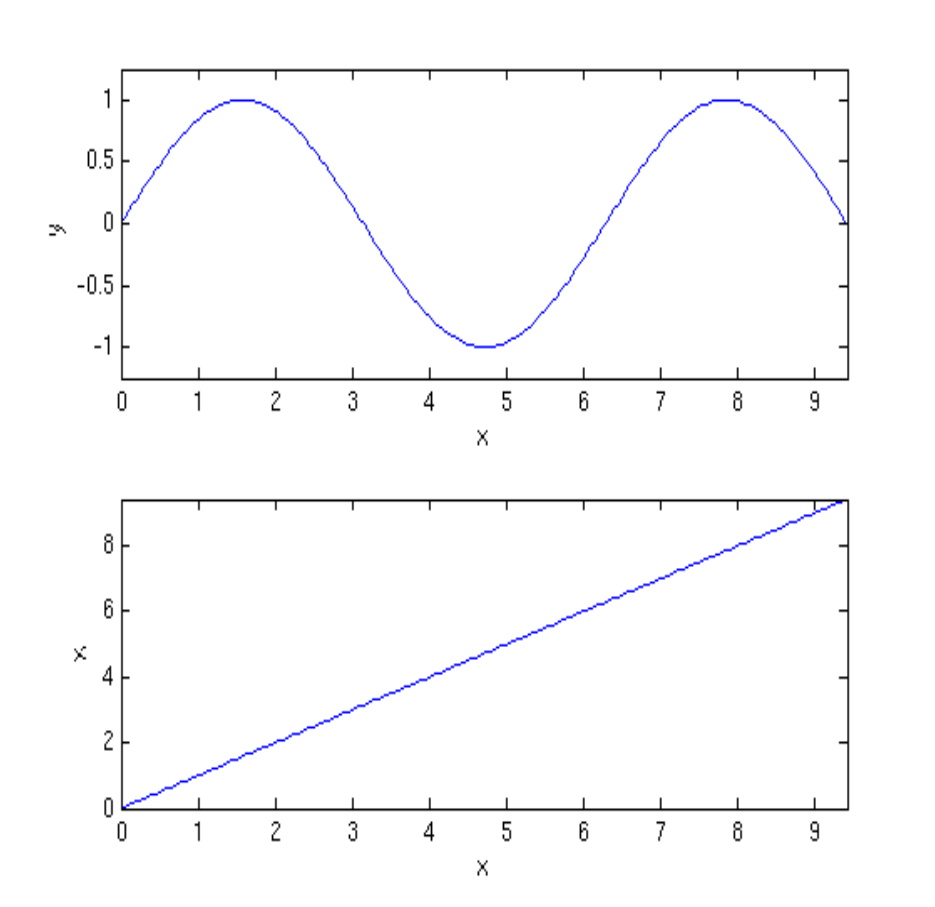
- General:
  - title('y = sin(x)');
  - xlabel('x'); ylabel('y');
  - axis([0 3*pi -1.25 1.25]);

Gráficas (II)



- Salvar la gráfica:
  - print -dpng sin
  - print(h,'-dpng','sin')

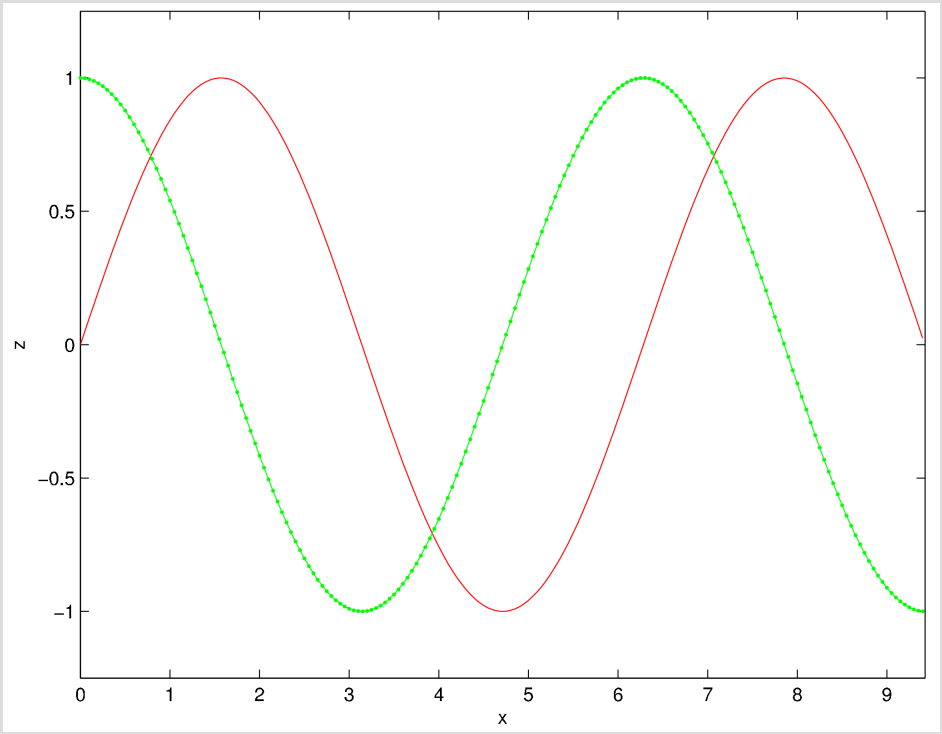
Gráficas (III)



- Subgráficas:

```
subplot(2,1,1);
plot(x,y);
axis([0 3*pi -1.25 1.25]);
xlabel('x'); ylabel('y')
subplot(2,1,2);
plot(x,x);
axis([0 3*pi 0 3*pi]);
xlabel('x'); ylabel('x')
```

Gráficas (IV)



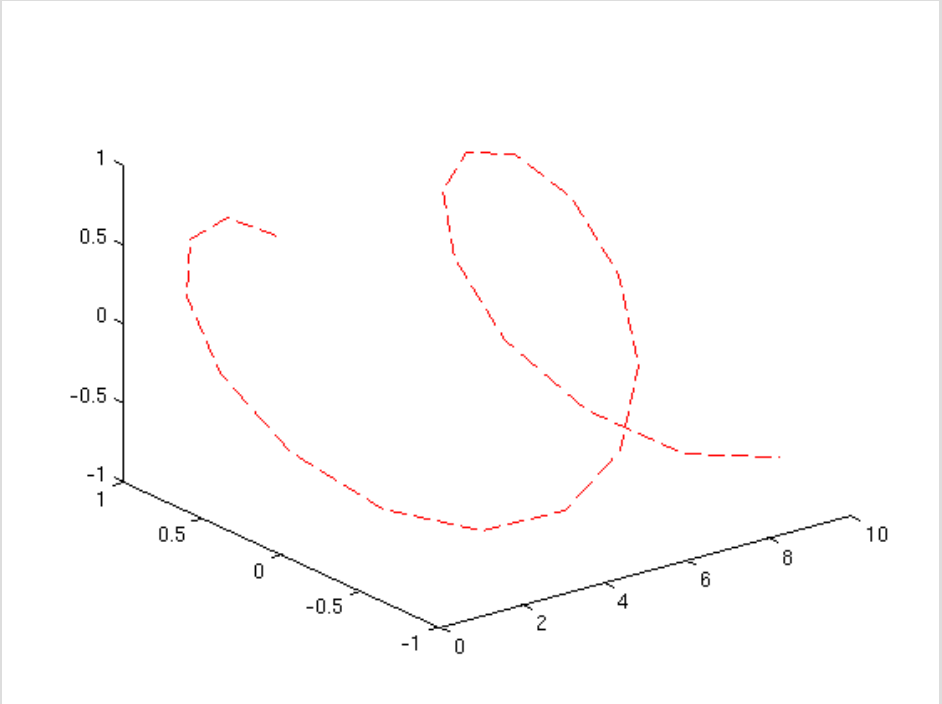
- Superposición:

```
plot(x,y,'r-');
axis([0 3*pi -1.25 1.25]);
xlabel('x'); ylabel('y');
hold on
z = cos(x);
plot(x,z,'g.-');
axis([0 3*pi -1.25 1.25]);
xlabel('x'); ylabel('z');
hold off;
```

Gráficas (V)

- Funciones de dos variables:

```
x = 0:0.5:10;
y = sin(x);
z = cos(x);
plot3(x,y,z,'r--');
```

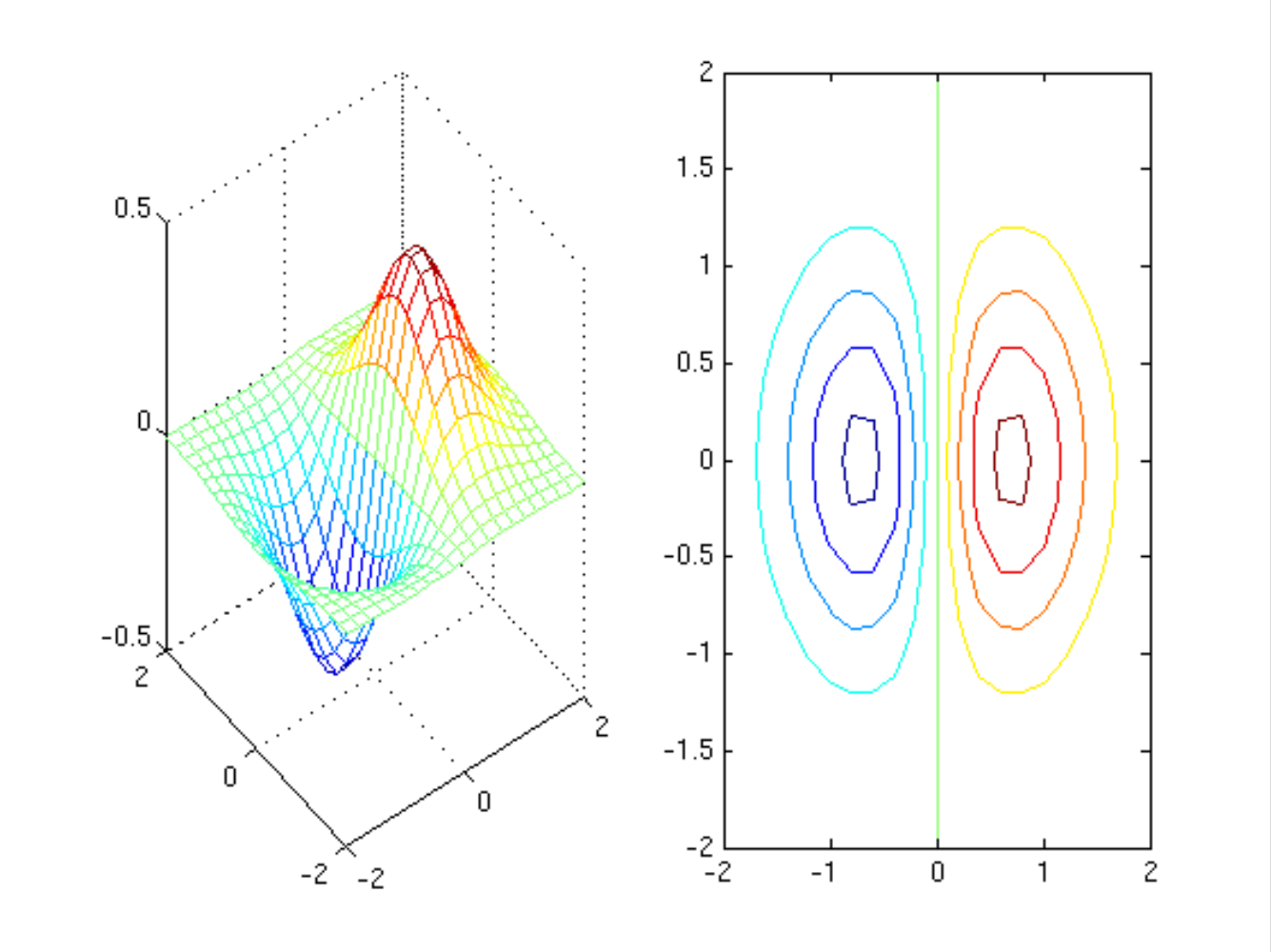


Gráficas (VI)

- Funciones de 2 variables:

```
x = -2:0.2:2;
y = -2:0.2:2;
[X,Y] = meshgrid(x,y);
Z = X .* exp(-X .^ 2 – Y .^2);
subplot(1,2,1);mesh(X,Y,Z);
subplot(1,2,2);contour(X,Y;Z);
```

Gráficas (VII)



Ejercicios MATLAB

2º Curso de II e ITIS
Curso 06/07

Marta Penas Centeno

Ejercicios con Matrices

- M1 = [1 2 3; 4 5 6; 7 8 9];
- M2 = 2 * ones(3,3);

  M1 + M2, M1 – M2, M1 * M2, M1 / M2, M1 \ M2,

  M1 .* M2, M1 ./ M2, M1 .\ M2

  M1^2, M1.^2

Graficos 2D

- Dibujar la funcion 'y = x * exp(x^2)' con x variando entre -5 y 5 a intervalos de 0.01. Salvar el resultado como grafica1.png.
- Dibujar en una sola ventana con dos subventanas las funciones 'y = sin(x)' y 'z = cos(x)' con x variando entre -5 y 5 a intervalos de 0.01. La primera en líneas rojas continuas, la segunda en lineas azules discontinuas.

Gráficos 3D

- Dibujar la función z = sin(x) + cos (y) donde x e y varían entre -5 y 5 a intervalos de 0.05.
  - Usar las funciones mesh, contour y plot3.

Neural Networks Toolbox

2º Curso de II e ITIS
Curso 06/07

Marta Penas Centeno

Funciones generales

- Error cuadrático medio:
  - mse(salida_deseada – salida obtenida);
- Máxima velocidad de aprendizaje:
  - maxlinlr(P [, 'bias']);
  - P es una matriz, cada columna de P es un vector de entrada
    - P = [e11 e12; e21 e22]
    - E1 = [e11; e21]
    - E2 = [e12; e22]

Regla delta (Widrow-Hoff)

- dW = learnwh(W,P,[],[],[],T,E,[],[],[],LP,[]);
  - W: matriz de pesos, cada fila representa los pesos de un elemento de procesado (SxR).
  - P: matriz de entrada, cada columna representa un vector de entrada (RxQ).
  - T: matriz de salidas deseadas, una salida deseada por columna (SxQ).
  - E: matriz de errores, un error por columna (SxQ).
  - LP: parámetros de aprendizaje.
    - LP.lr = learning rate
    - LP.mc = momentum constant
    - LP.dr = decay rate
  - dW: incremento de los pesos (SxR).

Creación de una red adaline

- NET = NEWLIN(PR,S,ID,LR);
  - PR: matriz con valores max y min que puede tomar una entrada.

    PR = [min1 max1; min2 max2; min3 max3]
  - S: número de elementos de procesado.
  - ID: retardo de la entrada, por defecto [0].
  - LR = learning rate.
  - NET: red adaline

    NET.biasConnect
    NET.trainParam.epochs
    NET.trainParam.goal
    NET.IW{1,1}
    NET.b{1,1}
    NET.inputWeights{1,1}.learnParam.lr

Entrenamiento estático (I)

- [net,TR,Y,E] = train(NET,P,T)
  - NET: red a entrenar.
  - P: vectores de entrada.
  - T: salidas deseadas.
  - net: red entrenada.
  - TR: training record (iteración + error)
  - Y: salida de la red
  - E: errores de la red

Entrenamiento estático (II)

- Dados los vectores de entrada E1 = [e11 e12], E2 = [e21 e22] con sus salidas deseadas T1 y T2.
- Entrada a la estructura:
  - Matriz donde cada columna es un vector de entrada: P= [e11 e21; e12 e22].
  - *Cell array* con un vector columna en cada posicion: P = {[e11; e12] [e21; e22]}
- Salida deseada:
  - Vector: T = [T1 T2];
  - Cell array: T = {T1 T2}

Ejecución estática

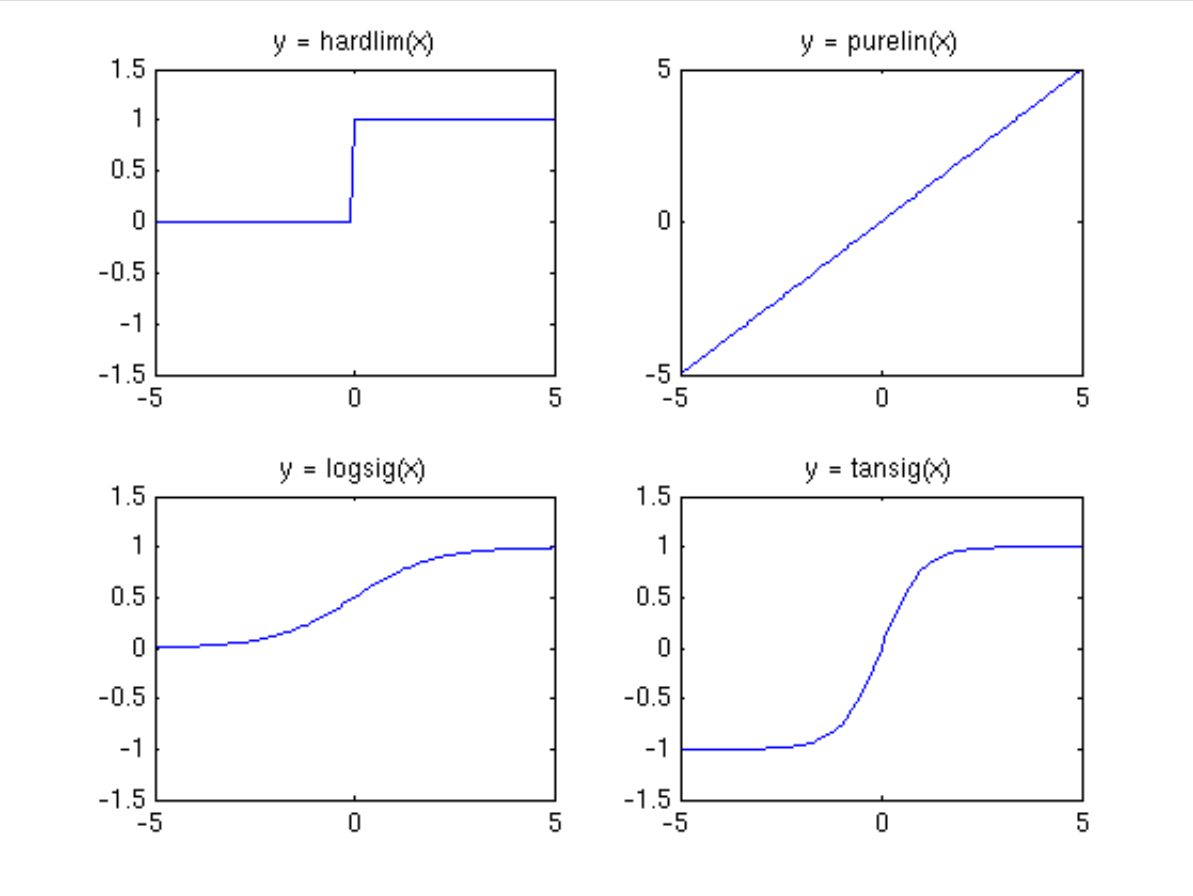
- [Y[, E]] = sim(NET,P[, T]);
  - NET: red ya entrenada.
  - P: vectores de entrada.
  - T: las salidas deseadas.
  - Y: salidas obtenidas.
  - E: errores.
  - Más información: help network/sim.

Entrenamiento adaptativo

- [net,Y,E,Pf,tr] = adapt(NET,P,T,Pi);
  - NET: red a entrenar.
    - NET.adaptParam.passes.
  - P: vectores de entrada.
  - T: las salidas deseadas.
  - net: red ya entrenada.
  - Y: salidas obtenidas.
  - E: errores obtenidos.

Funciones de transferencia

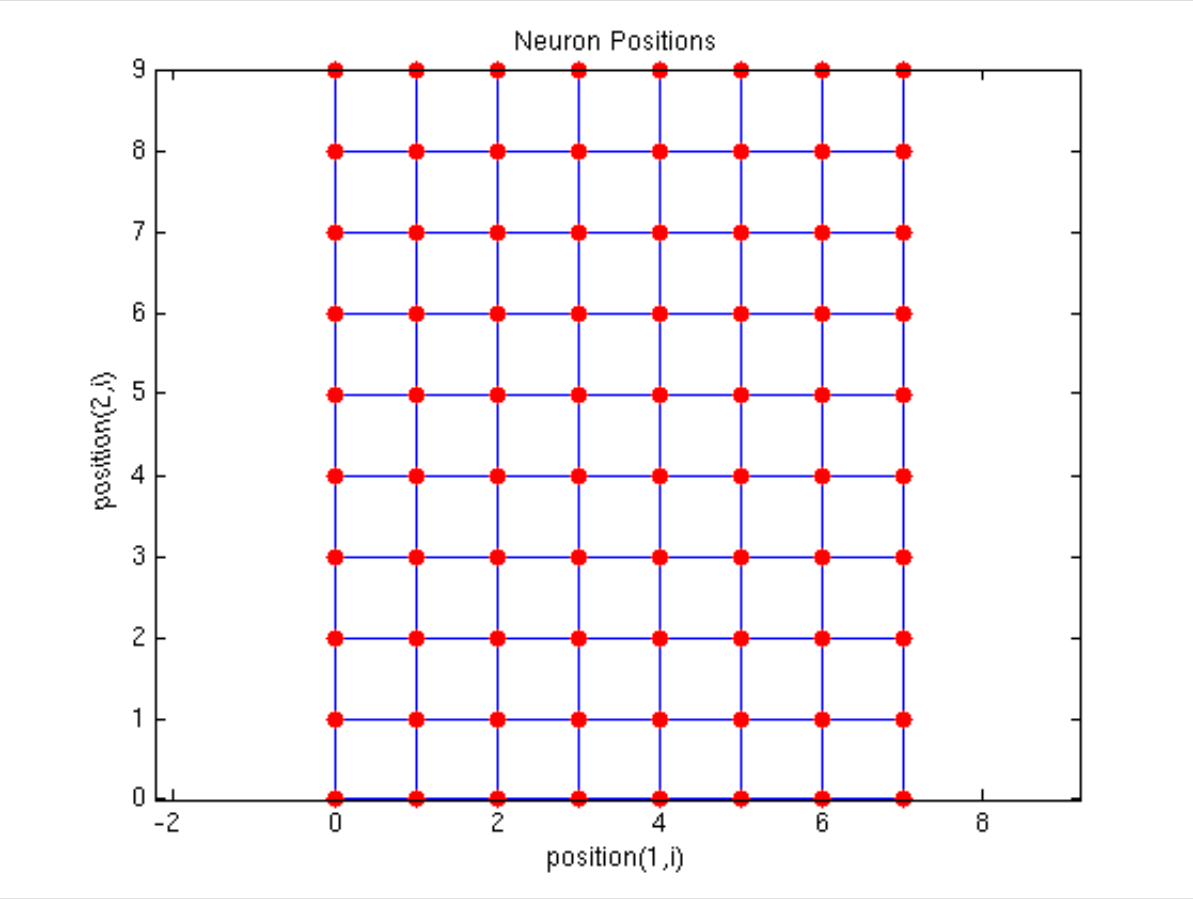
- net.layers{1,1}.transferFcn = 'funcion'



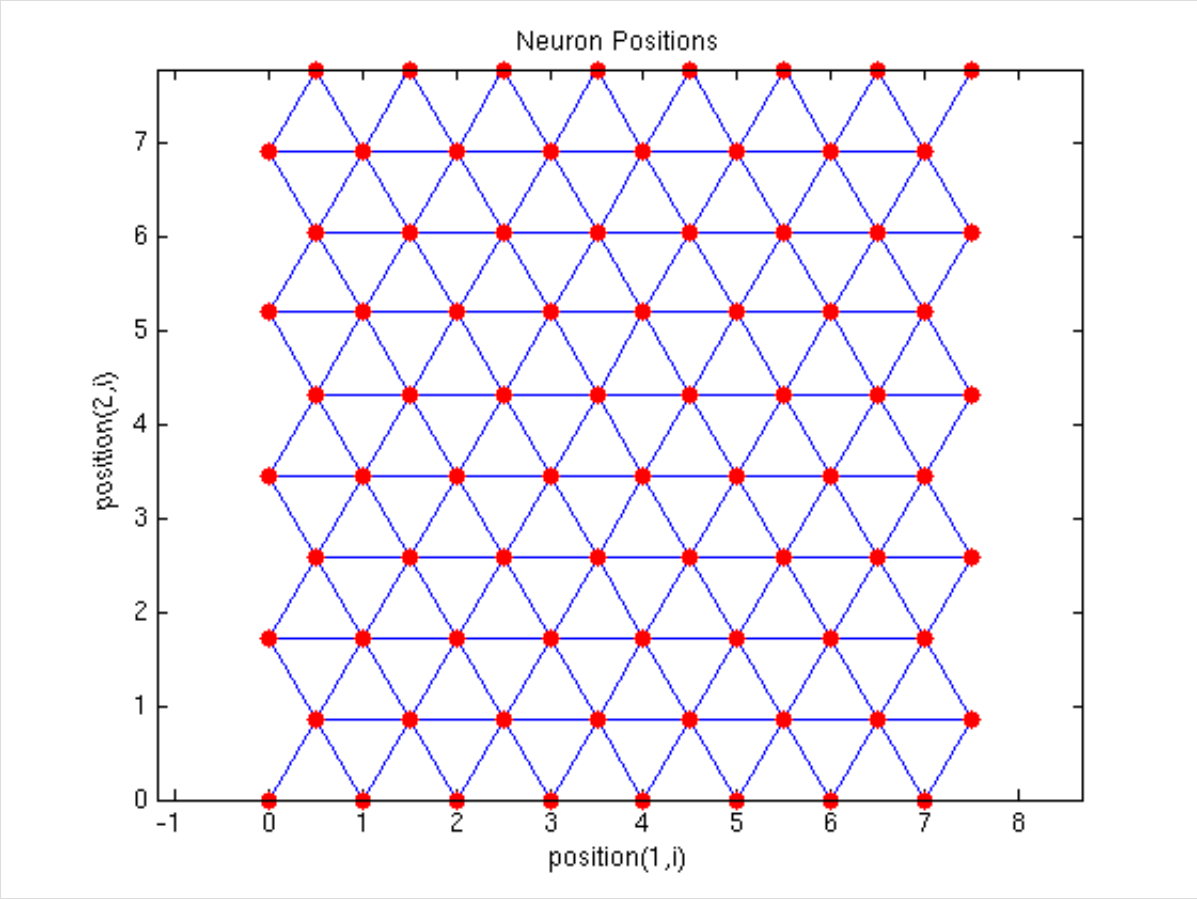
Creación de un SOM

- NET = NEWSOM(PR,[D1,D2,...],TFCN);
  - PR: matriz con los valore max y min de cada entrada.
  - Di: dimensión de la i-ésima capa [10 10].
  - TFCN: función de topología (hextop / gridtop / randtop).
  - NET: red SOM
    - NET.trainParam.goal
    - NET.trainParam.epochs
    - NET.trainParam.show

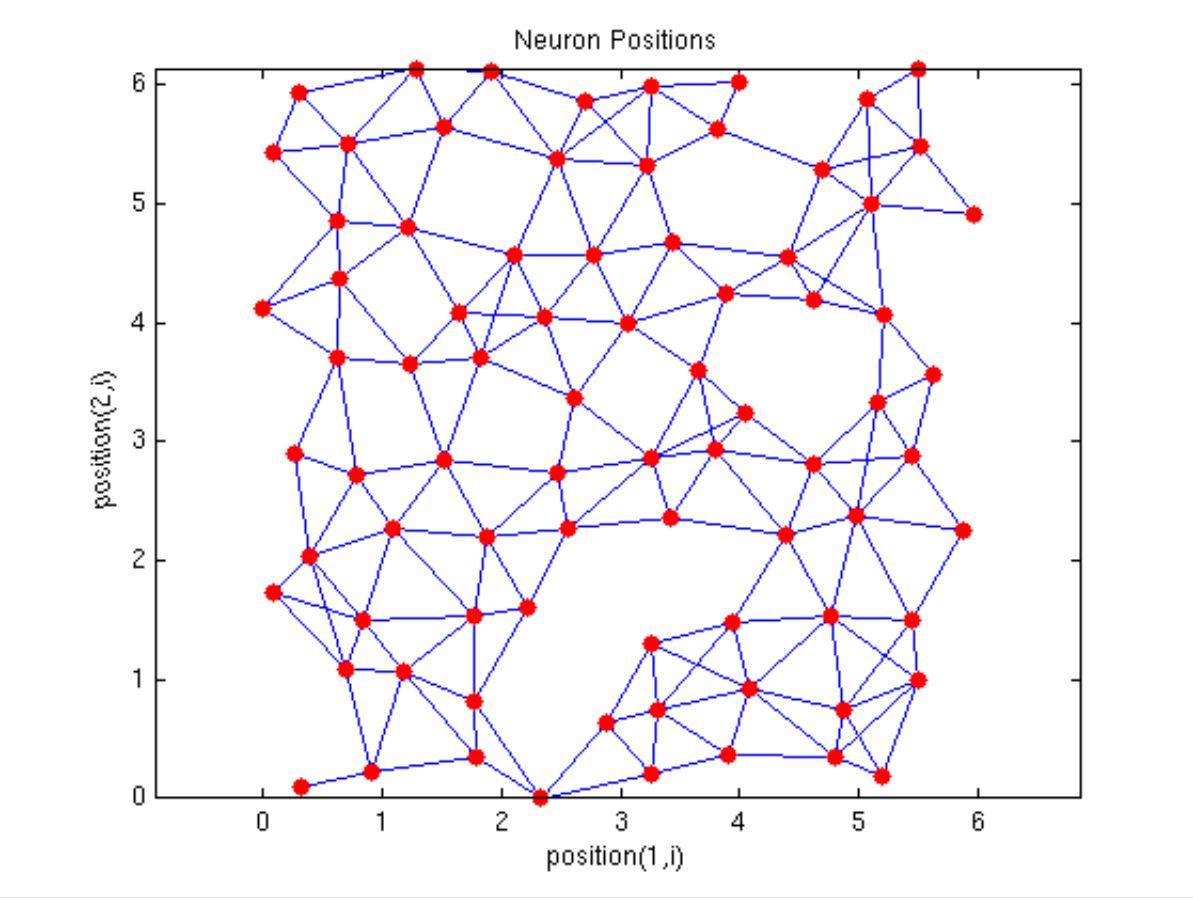
Topologías de un SOM (I)



Topologías de un SOM (II)



Topologías de un SOM (III)



Dibujar un SOM

- plotsom(pos)
  - pos: matriz con las posiciones de los elementos de procesado, net.layers{1}.positions.
- plotsom(W,d)
  - W: matriz de pesos, net.IW{1,1}
  - d: matriz de distancias, net.layers{1}.distances.