FUNCIONES BÁSICAS DE OCTAVE PARA ANÁLISIS NUMÉRICO

Los programas en Octave, se elaboran en documentos llamados Scripts. Tambien existen programas en las llamadas function, que son de uso común a varios programas (script), desde donde pueden ser llamados con distintos argumentos. De hecho, las funciones predefinidas del programa son function.
Normalmente un script, comienza con las siguientes 3 sentencias para poder trabajar de manera prolija:

clear : al ejecutarse el programa, borra la asignación de variables que se hicieron en anteriores ejecuciones.

close all : Cierra todos los gráficos de ejecuciones anteriores.

clc : Limpia la ventana de comando, de los mensajes anteriores.

; al terminar un renglón, evita que se muestre el resultado en la ventana de comandos.

VARIABLES Y VECTORES

Se puede definir el rango de una variable como
n = 0 : 0.1 : 1 . genera en n una variable, de 0 a 1 con intervalos de valor 0.1 De esta manera se ha generado un vector **n** con los valores de la variable.

Se pueden definir las funciones, con variables de rango como argumentos, generando un vector como resultado:

x = 2 * cos(2 * pi * n) . donde x es un vector con los valores de la función en cada valor de n definido en la variable de rango.
Esto es más eficiente que las definiciones con ciclos iterativos.

También podemos identificar elementos de un vector, usando índices:

a = x (2:6) . donde a, contiene los elementos del x entre 2 y 6.

Las siguientes funciones, generan vectores (o matrices), con componentes nulas o unitarias:

a = ones (1,5) . vector fila de 5 elementos unitarios.
b = zeros(5,1) . vector columna de 5 elementos nulos.

Los unos en el argumento indican que son vectores.

c = zeros (2,6) . matriz de 2x6 con elementos nulos.

Se pueden agrupar vectores para generan vectores compuestos :

b = [a , zeros(1,5), c] . b es un vector fila, compuesto los vectores filas entre corchetes separados por comas.

c = [a ; b] . c es una matriz, cuyas filas serían los vectores fila a y b. las , (coma) separan filas, los **; (punto y coma)** separan columnas en la definición del arreglo.

OPERACIONES MATRICIALES (Y VECTORIALES)

Recordemos que en la definición de las matrices, las **,** (coma), separan los elementos dentro de una misma fila, y los **;** (punto y coma) separan las distintas filas entre sí.

B = A' . B es la matriz o vector A transpuesto. En un vector, pasa de ser vector fila a columna o viceversa.

B = inv(A) . B es la matriz inversa de A.

C = A + B . C es la matriz o vector suma elemento a elemento de A y B.

C = A * B . C es el producto matricial de A y B. si A y B son vectores, uno debe ser fila y el otro, columna.

En las operaciones matriciales, usualmente al agregar un . (punto), la misma se realiza sobre los elementos del arreglo, de manera individual:

B = A .*5 = A * 5 . Los elementos de B, son los de A multiplicados por 5.

B = A .^2 . Los elementos de B, son los de A elevados al cuadrado.

C = A .*B . Los elementos de C, son los de A*B elemento a elemento.

z = conv (x,y) . el vector z, es la convolución de los vectores x e y.

Se puede agregar un tercer argumento para indicar la dimensión de z. Asi podemos tener :

z = conv (x,y.''full''). z tiene la dimensión completa, esta es la opción por default..

z = conv (x,y.''same''). z tiene la parte central de la convolución con la dimensión del vector x. (por ejemplo si x es la señal de entrada a un filtro, z es x filtrada.

X = fft(x) Calcula la transformada rápida de Fourier del vector x. Para la efectividad del algoritmo se requiere que los vectores tentan dimensión 2^n.

x = ifft(X) Calcula la transformada rápida de Fourier inversa del vector X de dimensión 2^n.

abs(z) Calcula el módulo del complejo z.

arg(z) Calcula el ángulo de fase en radianes del complejo z.

conj(z) Calcula el complejo conjugado z

real(z) . Obtiene la parte real de z.

imag(z) . Obtiene la parte imaginaria de z.

a=numel(b) Número de elementos del arreglo b.

a=length(b) Número de elementos del vector b.

a=max(b) Devuelve el máximo valor de los elementos del vector b.

[a,b]=max(x) Devuelve a = el máximo valor de los elementos del vector x. b = Posición de a.

.**x=randn(1,5)** vector de dimensión 5, con números aleatorios de distribución normal, con media = 0, y desvío standart = 1 (entre -3 y 3).

.**x=rand(1,5)** vector de dimensión 5, con números aleatorios de distribución uniforme entre 0 y1.

x=linspace(a,b) Vector fila con 100 puntos igualmente espaciados entre a y b.

x=linspace(a,b,N) Vector fila con N puntos igualmente espaciados entre a y b

x = hamming(M) Vector de dimensión M con la ventana de Hamming (simétrica).

GRAFICOS 2D : plot y stem.

Los gráficos, se van generando en distintas figuras. Las figuras, podemos numerarlas en el programa (script), pero si cada vez que grafiquemos, anteponemos la sentencia **figure()**, el programa automáticamente las genera numeradas secuencialmente, con cada gráfico.

plot(x) . grafica los valores de x. en la variable independiente coloca las posiciones i dentro del vector (x(i) vs i).

plot(x,y) . grafica los valores del vector y en función de los del vector x.

plot(-1:0.1:5,y) . Los valores del eje x, pueden ser definidos en un vector o directamente como variables de rango. Siempre deben coincidir en la cantidad de elementos.

Por default, el **plot** utiliza líneas continuas en los gráficos, se pueden agregar propiedades al mismo, como cambiar el tipo de línea, agregar marcadores en cada valor, cambiar el color, etc. ejecutando **help plot** en la ventada de comandos se puede ver con más detalle. Se pueden manejar dando los pares propiedad y valor, tantos como se quiera por ej.

plot(x,y,'linestyle','--','maker','o','markeredgecolor','g') . Este gráfico es de y vs x, en línea discontinua, con marcadores tipo círculos llenos en cada punto y de color verde.

Una manera simplificada de definir las características gráficas es llamada fmt format, que tiene una cadena de 4 caracteres opcionales, con las siguientes características en su sintaxis:

plot(x,y,'-- o g ; señal de entrada;') en la cadena de caracteres entre apóstrofes **' ',** el primer valor es el estilo de línea, el segundo el del marcador (si no se coloca nada, no grafica marcadores), el tercero es el color (si no se coloca pone uno por default, el último es si se quiere agregar una legenda, va encerradas con **;** (punto u coma). Si no se especifica el estilo de línea, sólo grafica los marcadores, como sería la representación de señales en TD, pero para eso es más práctico el uso de la función **stem**.

Para representara varias funciones en el mismo gráfico, se colocan todos los pares x,y uno a continuación de otro

plot(x1,y1,'-- o g ; señal de entrada;',x2,y2,x1,y3) . Aquí el gráfico contiene 3 funciones. La condición es que sean de igual dimensión en cada par.

Si varias funciones se representan para la misma variable independiente, éstas se pueden colocar como filas o columnas de una matriz (primero prueba las columnas).

plot(x,[y1;y2]) , legend (' función y1',' función y2') . Representa las funciones y1 e y2 vs x, con sus respectivas legendas.

La función **stem** tiene características similares al **plot,** sólo no grafica la curva continua, sino los valores discretos. Una diferencia en la representación de varias

funciones en el mismo gráfico, es que si la variable dependiente es una matriz, grafica el vector x, vs cada columna de la matriz, para comparar con el ejemplo de plot anterior en el stem quedaría :

stem(x,[y1;y2]') , legend (' función y1',' función y2') . Representa las funciones y1 e y2 vs x, con sus respectivas legendas. Notar que la matríz ha sido transpuesta, para que cada función sea una columna de la misma. Se pueden agregar propiedades:

stem(x,[y1;y2]','color','r','markeredgecolor','g','marker','^') , legend(' función y1',' función y2') , xlabel('tiempo ') , ylabel(' función y1','función y2'). Donde además de las propiedades, se agregan leyendas para los ejes y las funciones.

stem(x,[y1;y2]',' -- ^ r ') . aquí se le dan las propiedades de líneas verticales discontinuas, marcadores triangulares, todo en rojo.

REPRESENTACIÒN DE VARIOS GRÁFICOS EN LA MISMA FIGURA:

Para esto, existe la sentencia **subplot,** que divide la figura en filas y columnas indicadas en los dos primeros índices, el último índice indica cuál se está representando por Ej.:

Subplot(3,1,1) . Divide la figura en tres filas de una columna, y representamos la primera fila con por ej. **stem(x,[y1;y2]',' -- ^ r ')**

xlim([a,b]) ; Los valores adoptados por las variables representadas en x van de a hasta b, si se coloca en todos los subplot, los gráficos tendrán la misma escala de representación en x.

Subplot(3,1,2) define la segunda fila de la figura

stem(x,[a1;a2]',' -- ^ r ')

xlim([a,b])

Subplot(3,1,3) define la tercer fila de la figura

stem(x,[b1;b2]',' -- ^ r ').

xlim([a,b])

CICLOS :

for i=1:5

sentencias

endfor

CONDICIONAL IF

If (condición lógica)

sentencias

else o else if

endif

EXISTEN OTROS COMO WHILE Y SWITCH

FUNCTION genera subprogramas, que se invocan desde otro programa principal o script.

function y = convolución (x,h) **%** al ser llamada, con los argumentos x y h,

sentencias **%** devuelve como resultado la convolución

endfunction **% y**

LLAMADO ; **y = convolución (x,h)** desde un scritp.

FUNCTION ANÓNIMA Se puede definir, en cualquier lugar del script.

Ejemplo : escalón unitario :

U = @ (n) (n ≥0) ; definimos una function anónima, donde n es un vector argumento y U el vector generado, con unos si es verdadera la proposición, y ceros si es falsa.

Aplicación : n=-10:10 (definimos un vector que es una variable de rango, con valores de -10 a 10.

a=U(n+1), a es el escalón desplazado entre -10 y 10.