Introducción:

- MatLab es el nombre abreviado de “Matriz Laboratory”.
- Es un programa que se encuentra estructurado en forma matricial, permitiéndonos realizar cálculos numéricos de un número grande de datos.
- Puede trabajar vectores y matrices, sin la necesidad de declarar librerías; además de trabajar en el campo real y complejo, variables simbólicas, entre otros.

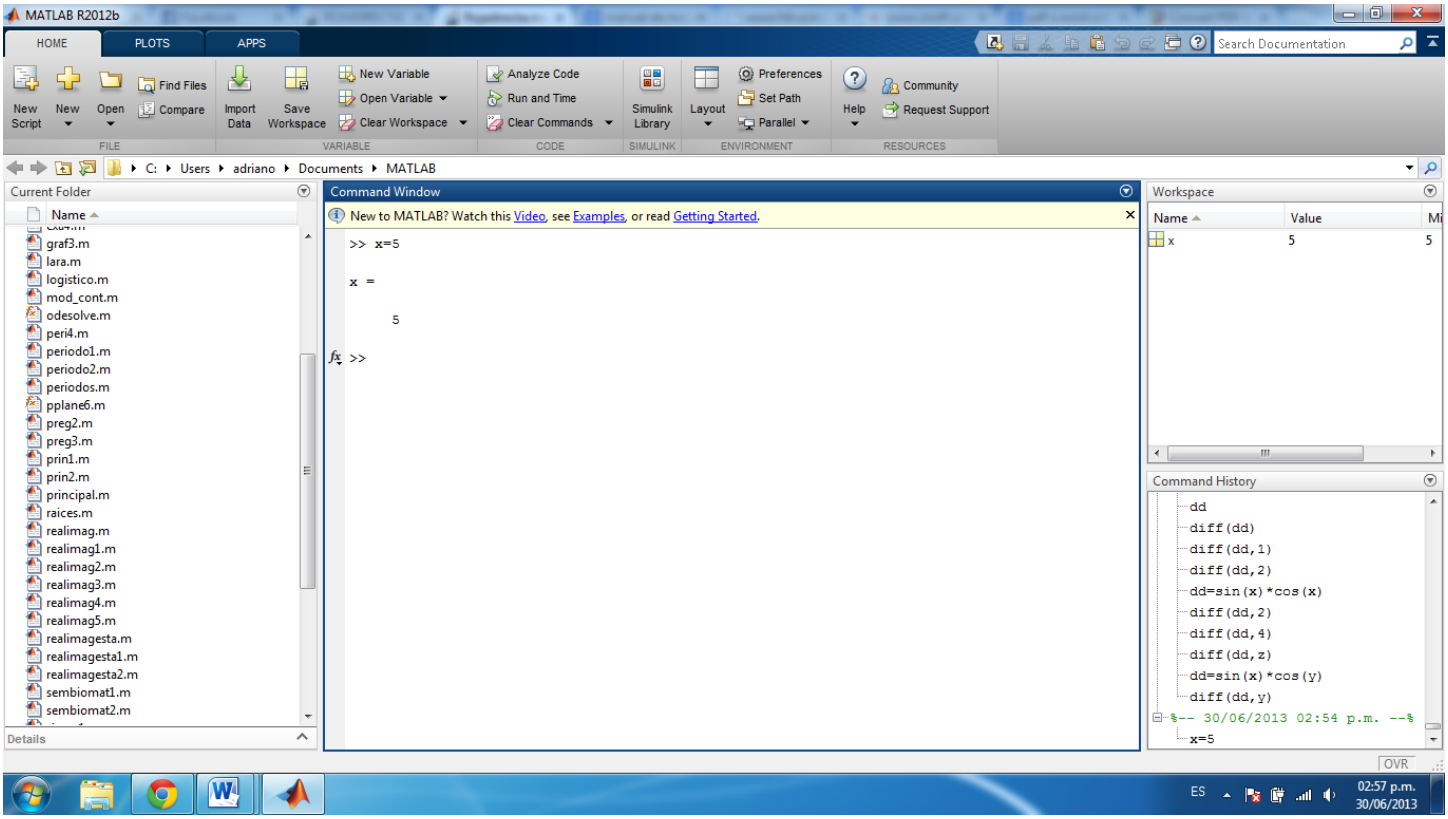
Pro’s:

- Es un lenguaje amigable de alto rendimiento para cálculos técnicos.
- Maneja cálculo matricial, álgebra lineal, polinomios, ecuaciones diferenciales ordinarias, gráficas, etc.
- Tiene comandos preestablecidos, los cuales simplifican mucho el trabajo.
- Permite trabajar con una gran cantidad de datos.

Contra’s:

- No es un software libre.
- Es muy pesado.
- Las variables no tienen tipo de almacenamiento definido, lo cual causa el uso de más espacio en el guardado.

Entorno gráfico:



- Command Window: Es la ventana de mayor tamaño en el entorno de Matlab, en la cual casi siempre desarrollaremos nuestro trabajo.
- Workspace: En esta ventana observaremos la información sobre las variables que estamos ocupando en el momento.
- Command History: Es la ventana que será como nuestro borrador, ya que en esta aparecerá todo lo que hemos ido trabajando en la ventana de comandos.
- New Script: Opción en donde podremos trabajar nuestros programas propios, los cuales podrán ser guardados y utilizados más de una vez.
- Current folder: Espacio en donde aparecerá los programas hechos en la ventana Script, los cuales fueron guardados.

Operaciones Básicas:

Matlab ofrece la posibilidad de realizar las siguientes operaciones básicas:

Operación	Símbolo	Expresión en Matlab
suma	+	a + b
resta	-	a - b
multiplicación	*	a * b
división	/	a / b
potencia	^	a ^ b

- Hay que tener en cuenta que Matlab no tiene en cuenta los espacios entre los signos de las operaciones básicas.
- Los niveles de jerarquía en las operaciones, son los mismos que se usan en la vida matemática.
- El punto y coma al final de una operación, le indica al Matlab, que haga la operación, pero que no la muestre en pantalla.
- Hay que tener en cuenta que Matlab sólo trabaja con variables que se encuentren definidas previamente, y estas son leídas de arriba hacia abajo.

Algunos Comandos Básicos:

- Clear: Borra específicamente la variable que se desea… clear x
- Clear All: Borra todas las variables que hayan sido definidas previamente.
- Clc: Limpia la ventana de comandos, lo cual no significa que borre las variables.
- Format short: Muestra el valor de la variable con cuatro dígitos luego del punto decimal.

$$pi \approx 3.1416$$

- Format long: Muestra el valor de la variable con quince dígitos luego del punto decimal.

$$pi \approx 3.141592653589793$$

- Disp( ): Comando que imprime lo que se encuentra dentro del paréntesis, en una forma muy simple.

$$disp(pi) = 3.1416$$
$$disp('hola') = hola$$
$$disp(3) = 3$$

- Fprintf( ): Comando que imprime lo que se encuentra dentro del paréntesis, en un estilo más definido en relación al comando disp.

$$fprintf('\%4.6f', pi) = 3.141593$$
$$fprintf('el\ valor\ de\ pi\ es\ \%4.6', pi) = el\ valor\ de\ pi\ es\ 3.141593$$
$$fprintf('\ \backslash t\ el\ valor\ de\ \backslash t\ pi\ es\ \%4.6', pi) = \quad el\ valor\ de \quad pi\ es\ 3.141593$$

Comando	Descripción
\t	Tabulación
\n	Enter
\b	Retrocede un espacio
\f	Tabulación más pequeña

- Input( ): Comando que se utiliza para asignar valor a una variable, esto mediante ingreso.

$$x = input('ingrese\ el\ valor\ de\ x')$$

- i , j: Variables que si no son definidas previamente, tienen por default el valor $\sqrt{-1}$, es decir, $0 + i = 0 + j$.
- % : El signo tanto por ciento es utilizado para poner un comentario.

$$\%\ comentario$$

Nota:

- Cuando se genera algún ciclo infinito, lo cual ocurre en los bucles, se teclea $ctrl + c$, lo cual para el ciclo.
- Matlab opera por default en radianes.
- Date, clock y calendar son expresiones ya reservadas, para la fecha, hora y calendario respectivamente.

Algunas funciones matemáticas especiales:

- $abs(x)$: Valor absoluto o magnitud, si el número es complejo, de x.
- $sqrt(x)$: Raíz cuadrada de x.
- $log(x)$: Logaritmo natural de x.
- $exp(x)$: Exponencial de x.
- $sin(x)$: Seno de x.
- $cos(x)$: Coseno de x.
- $tan(x)$: Tangente de x.
- $cot(x)$: Cotangente de x.
- $sec(x)$: Secante de x.
- $csc(x)$: Cosecante de x.
- $sinh(x)$: Seno hiperbólico de x.
- $cosh(x)$: Coseno hiperbólico de x.
- $tanh(x)$: Tangente hiperbólica de x.
- $sind(x)$: Seno de x, cuando está en grados.
- $cosd(x)$: Coseno de x, cuando está en grados.
- $tand(x)$: Tangente de x, cuando está en grados.
- $cotd(x)$: Cotangente de x, cuando está en grados.
- $secd(x)$: Secante de x, cuando está en grados.
- $cscd(x)$: Cosecante de x, cuando está en grados.
- $log2(x)$: Logaritmo en base 2 de x.
- $log10(x)$: Logaritmo en base 10 de x.
- $real(x)$: Muestra la parte real de x.
- $imag(x)$: Muestra la parte compleja de x.
- $ceil(x)$: Redondea x hacia el infinito.
- $fix(x)$: Redondea x hacia cero.
- $floor(x)$: Redondea x hacia menos infinito.
- $round(x)$: Redondea x hacia el entero más próximo.
- $sign(x)$: Devuelve $-1, 0\ y\ 1$ si x es negativo, cero y positivo respectivamente.
- gcd(m, n): Máximo común divisor entre $m\ y\ n$.
- $lcm(m, n)$: Mínimo común múltiplo entre $m\ y\ n$.
- $rem(m, n)$: Resto de la división entre $m\ y\ n$.
- $nthroot(x, n)$: $\sqrt[n]{x}$
- $mod(m, n)$: $abs(floor(\frac{m}{n}))$
- $complex(x, y)$: Genera el número complejo $x + iy$.
- $conj(x)$: Genera el conjugado de x.
- $angle(x)$: Ángulo (en radianes) del número complejo x.
- $isreal(x)$: Devuelve 1 si x es real y 0 si es complejo.

Vectores:

- Matlab tiene la ventaja de crear vectores y matrices sin antes definir librerías como en otros programas.
- La forma de definir vectores es muy simple, tan sólo bastará encerrar los números entre corchetes y separar los números por un espacio, comas o una combinación de ellos.

$$x = [0\ 1\ 2\ 3\ 4]$$
$$x = [0,1,2,3,4]$$
$$x = [0,1\ 2,3\ 4]$$

- Si deseamos conocer algún(os) elementos específicos del vector, tan sólo se escribirá:
  - $x(i)$: Nos da la posición i del vector x.
  - $x(i:j)$: Mostrará los elementos desde la posición i hasta la posición j del vector x.
  - $x(i:j:k)$: Mostrará los elementos desde la posición i hasta la k, incrementándose en j, del vector x.
  - $x([i\ j\ k])$: Mostrará los elementos en la posición i, j y k del vector x.

- Otras formas de crear vectores, son las siguientes:

  $x = i:j:k = [i \quad i+j \quad i+2j \quad \ldots \quad k]$

  $x = linspace(a, b)$: vector que inicia en a, termina en b y tiene 100 elementos igualmente espaciados.

  $x = linspace(a, b, c)$: vector que inicia en a, termina en b y tiene c elementos igualmente espaciados.

- Es claro que para la suma y resta de vectores, estos deben tener la misma cantidad de elementos; en el caso de la multiplicación de vectores es claro que solo puede aplicarse el elemento a elemento, lo cual queda expresado por $.*$ (lo mismo ocurre con la división $./$ y la potencia $.\hat{}$).

Matrices:

- Para definir matrices en Matlab, se sigue el mismo criterio que con vectores, agregando el hecho de que al ingresar las filas, el salto queda expresado por el punto y coma.

$$x = [1\ 2\ 3\ ;\ 4\ 5\ 6\ ;\ 8\ 4\ 2]$$

- Debe quedar en claro que las filas son separadas por el punto y coma, además que deben tener la misma cantidad de elementos.
- Matlab tiene algunos comandos para generar matrices especiales, los cuales son:

  Sea v un vector:
  - $diag(v)$: genera una matriz diagonal, donde los elementos de esta son los elementos de v.
  - $toeplitz(v)$: genera una matriz simétrica de diagonal constante, donde v es la primera fila y columna.
  - $ones(n)$: genera una matriz de unos de dimensión $n \times n$.
  - $zeros(n)$: genera una matriz de ceros de dimensión $n \times n$.
  - $eye(n)$: genera una matriz identidad de dimensión $n \times n$.
  - $rand(n)$: genera una matriz de $n \times n$ con elementos de valor aleatorio entre 0 y 1 (distribución uniforme).
  - $randn(n)$: genera una matriz de $n \times n$ cuyos elementos siguen una distribución normal (media 0 y varianza 1).
  - $ones(m, n)$: genera una matriz de unos de dimensión $m \times n$.
  - $zeros(m, n)$: genera una matriz de ceros de dimensión $m \times n$.
  - $rand(m, n)$: genera una matriz de $m \times n$ con elementos de valor aleatorio entre 0 y 1 (distribución uniforme).
  - $size(A)$: dimensión de la matriz A.

Cambiar los elementos de una matriz conocida:

Sea $A \in M_{n \times m}$:

- $A(i, j) = 2$: Pone en la posición (i, j) el valor igual a 2.
- $A(i, :) = v$: Reemplaza los valores de la fila i por el vector v.
- $A(:, j) = v$: Reemplaza los valores de la columna j por el vector v.

Elementos de una matriz:

- $A(i,j)$: Nos muestra los elementos de la posición (i,j) de la matriz A.
- $A(i,:)$: Nos muestra toda la $i-esima$ fila de la matriz A.
- $A([i\ \ j]\ ,\ :)$: Muestra las filas $i\ ,\ \ j$ como una submatriz de A.

Operaciones con matrices:

- $A * B$: Multiplicación clásica entre dos matrices.
- $A.* B$: Multiplicación elemento a elemento entre dos matrices.
- $inv(A)$: Inversa de la matriz A.
- $pinv(A)$: Pseudoinversa de la matriz A.
- $\det(A)$: Determinante de la matriz A.
- $rank(A)$: Rango de la matriz A.
- $trace(A)$: Traza de la matriz A.
- $triu(A)$: Vuelve la matriz A en una triangular superior.
- $tril(A)$: Vuelve la matriz A en una triangular inferior.
- $eig(A)$: Muestra los valores propios de la matriz A.
- $poly(A)$: Devuelve el polinomio característico de la matriz A.
- $[L,U] = lu(A)$: Produce la factorización LU de la matriz A.
- $[S,E] = eig(A)$: Devuelve una matriz diagonal (E) de valores propios, y la de vectores propios (S).

Gráficas:

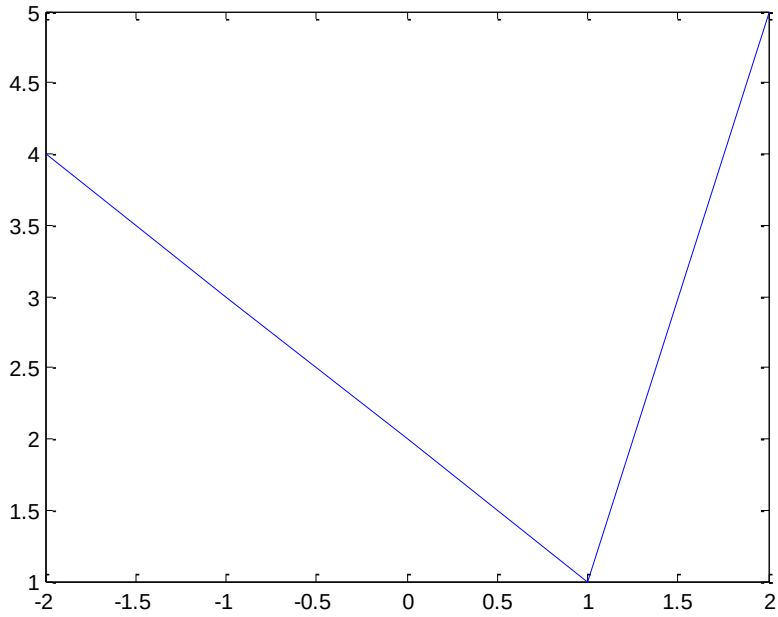
2-D:

El comando principal, o más básico, para hacer gráficas es el $plot(\,)$, el cual tiene la siguiente sintaxis:

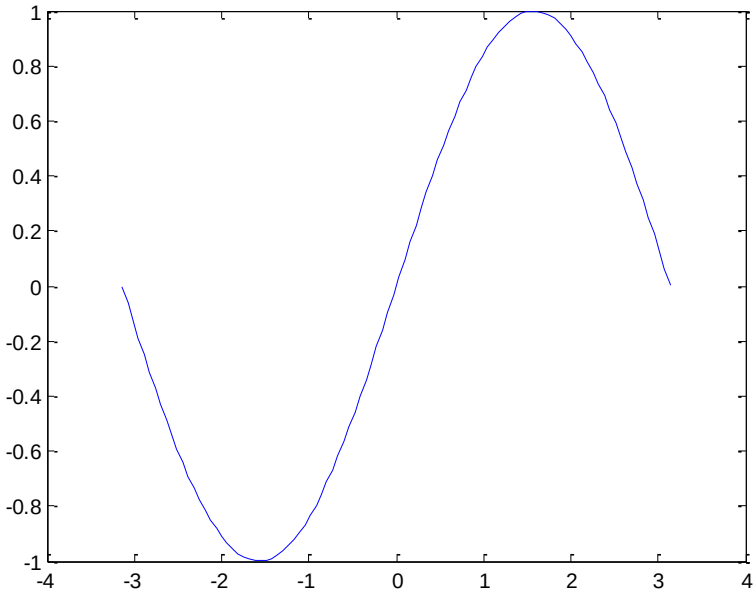
$$plot(x, y)$$

Donde x, y son vectores de la misma dimensión.

$x = [-2\ -1\ 0\ 1\ 2]$
$y = [4\ 3\ 2\ 1\ 5]$
$plot(x, y)$



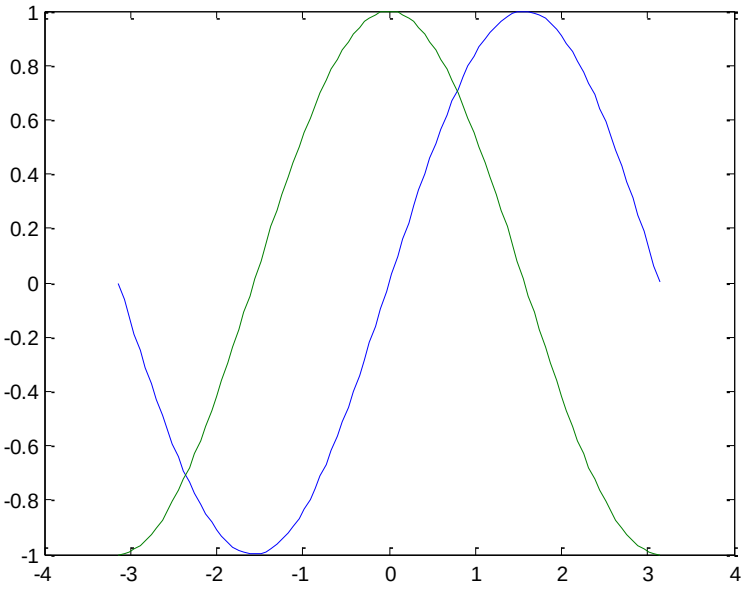
$x = linspace(-pi, pi)$
$y = \sin(x)$
$plot(x, y)$



La idea del gráfico en Matlab es que cada par ordenado $(x(i), y(i))$ sean unidos mediante rectas, lo cual es claro que mientras más puntos presente la función, más suave será esta.

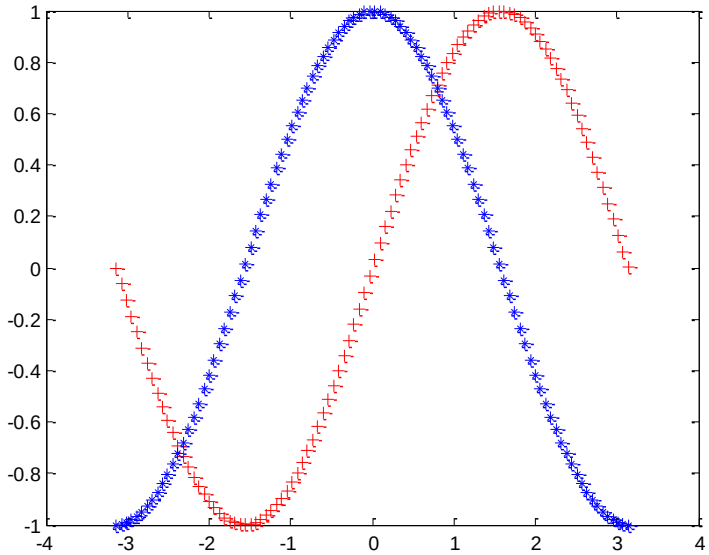
En Matlab se puede expresar dos gráficas al mismo tiempo, lo cual se expresa así:

$x = linspace(-pi, pi)$
$y = \sin(x)$
$y1 = \cos(x)$
$plot(x, y, x, y1)$



Matlab nos ofrece más colores para poder graficar, además de otro estilos en los puntos, lo cual se define en comando de la siguiente forma:

$x = linspace(-pi, pi)$
$y = \sin(x)$
$y1 = \cos(x)$
$plot(x, y, 'r+', x, y1, 'b*')$



Pero no son las únicas formas y colores:

Símbolo	Color
y	Amarillo
m	Lila
c	Turquesa
r	Rojo
g	Verde
b	Azul
w	Blanco
k	Negro

Símbolo	Estilo de línea
.	Punto
o	Circulo
x	Aspa
+	Mas
*	Asterisco
-	Línea
:	Dos puntos
-.	Línea y punto
--	Doble línea

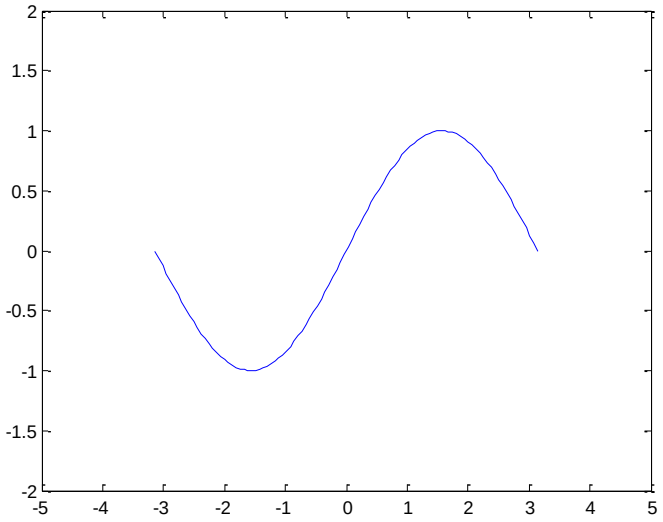
También podemos cambiar la escala de los ejes (uno o ambos), esto al cambiar el comando $plot$, por:

- $loglog$: Escala logarítmica para ambos ejes.
- $semilogx$: Escala logarítmica para el eje x y lineal para el eje y.
- $semilogy$: Escala logarítmica para el eje y y lineal para el eje x.

Otro de los inconvenientes que se presenta al graficar, es el tamaño de los ejes, los cuales se encuentran definidos como los máximos y mínimos valores que se designan a las variables.

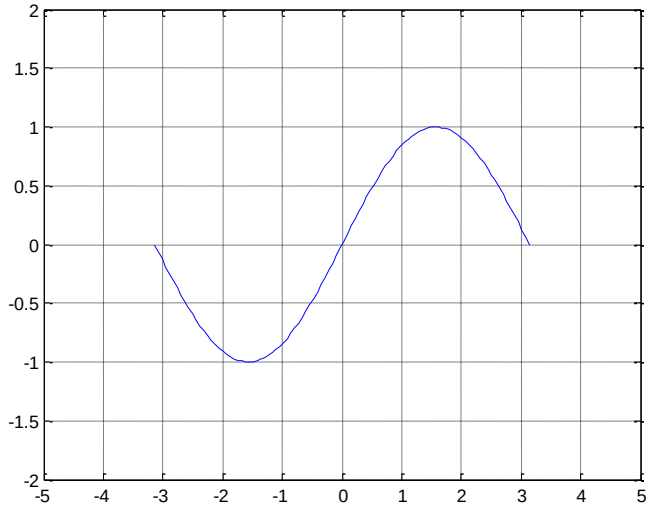
Pero esto puede ser reformulado, para lo cual está el comando $axis$

$x = linspace(-pi, pi)$
$y = \sin(x)$
$plot(x, y)$
$axis([-5\ \ 5\ \ -2\ \ 2])$



Asimismo, si deseamos poner una rejilla para que nuestra gráfica se aprecie mejor, está el comando $grid\ on$, y para que se vaya la rejilla está el comando $grid\ off$.

$x = linspace(-pi, pi)$
$y = \sin(x)$
$plot(x, y)$
$axis([-5\ \ 5\ \ -2\ \ 2])$
$grid\ on$



Otros comandos no menos importantes, son las llamadas etiquetas o referencias:

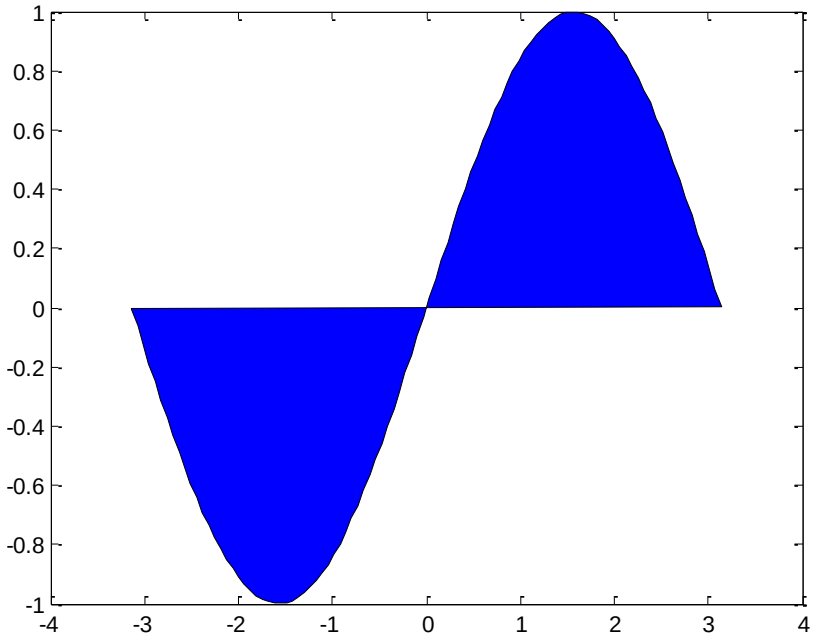
- $xlabel('texto')$: etiqueta sobre el eje x.
- $ylabel('texto')$: etiqueta sobre el eje y.
- $title('texto')$: título general.
- $text(x, y, 'texto')$: texto en el lugar especificado por las coordenadas (x, y).
- $gtext('texto')$: análogo a lo anterior, pero el lugar es especificado con el mouse.
- $axis\ equal$: escala de los ejes iguales.
- $axis\ square$: vuelve la gráfica un cuadrado, lo cual no significa que los ejes sean iguales.
- $axis\ normal$: desactiva los dos comandos anteriores.
- $legend('fig1', 'fig2', \dots, 'fig\ n')$: Muestra específicamente la leyenda de cada figura.

Ahora, si tenemos una gráfica ya presentada y queremos agregar otra, Matlab también tiene esta opción, la cual se desarrolla con el comando $hold\ on$.

$x = linspace(-pi, pi)$
$y = \sin(x)$
$plot(x, y)$
$hold\ on$
$y1 = \sin(x) * \cos(x)$
$plot(x, y1)$

Como recordamos, Matlab une los puntos mediante líneas, entonces, si queremos graficar alguna figura y más aún, darle algún relleno, cambiamos el comando $plot$, por el comando $fill$, lo cual pinta la parte interior de la gráfica que se desea.

$x = linspace(-pi, pi)$
$y = \sin(x)$
$fill(x, y, 'b')$



- $fplot$: Dibuja la gráfica de una función ya establecida.

  $fplot('funcion', [a\ b])$
  $fplot('funcion', [a\ b]\ , n)$, n es el número de puntos.

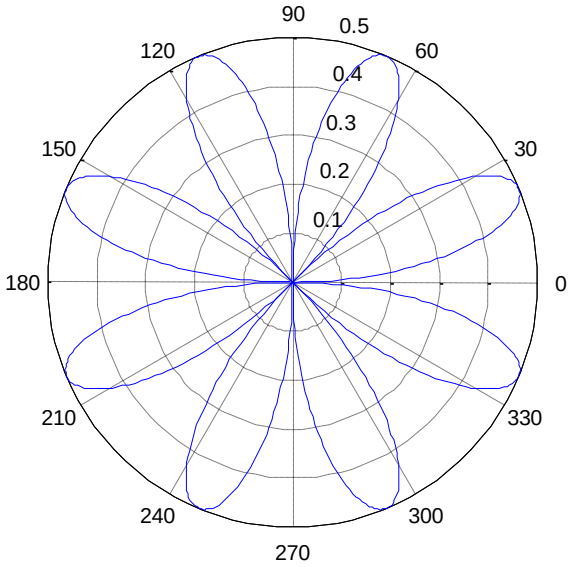
  $fplot('sin', [0\ pi])$
  $fplot('tanh', [-2\ 2])$

- $polar$: Grafica en coordenadas polares.

  $fplot(angulo\ , radio)$
  $fplot(angulo\ , radio\ ,'\ tipo\ de\ linea')$

  Ejemplo:

  $x = 0{:}\ 0.01{:}\ 2 * pi$
  $y = \sin(2 * pi)\ .* \cos(2 * pi)$
  $polar(x, y)$



Otra de las ventajas que tiene Matlab con respecto a las gráficas, es que se puede poner distintas gráficas en una sola ventana, pero en distintas posiciones, para lo cual está el comando $subplot$, el cual tiene la siguiente sintaxis:

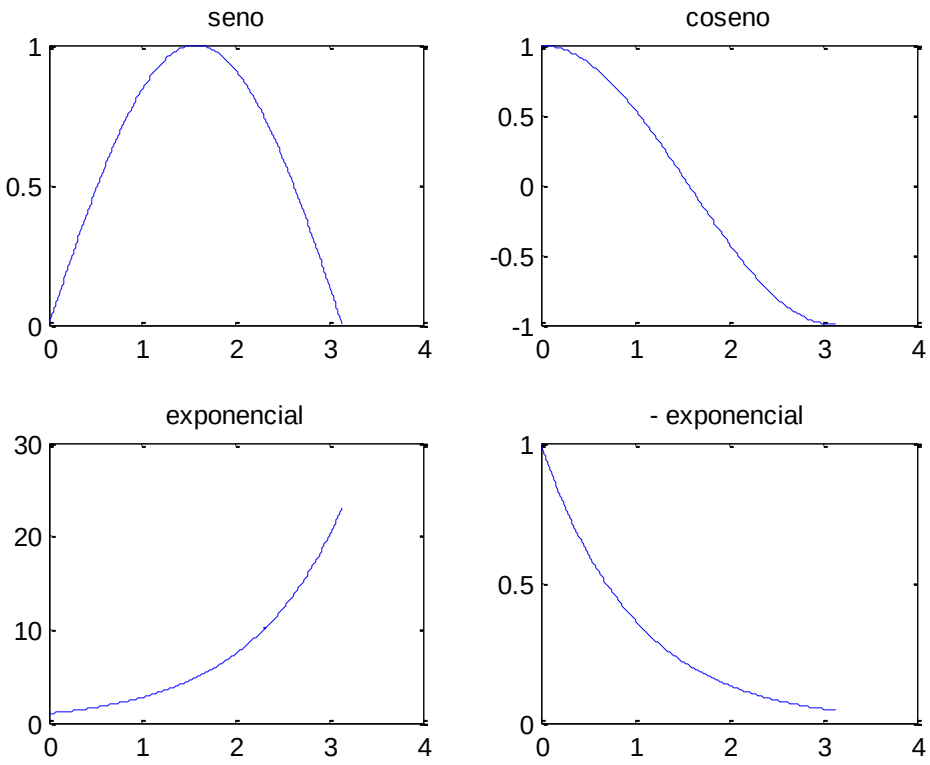
$$subplot(m, n, p)$$

Donde:

- m: número de filas.
- n : número de columnas.
- p : posición

Ejemplo:

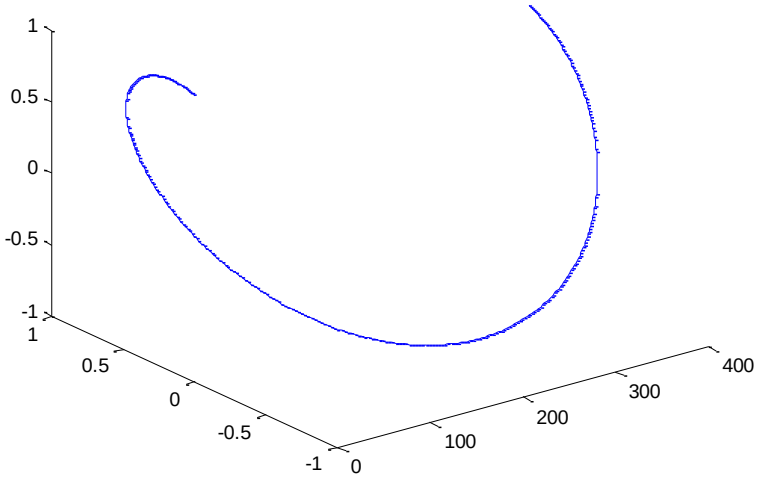
$t = 0{:}\ 0.01{:}\ pi$
$x = \sin(t)$
$y = \cos(t)$
$z = \exp(t)$
$w = \exp(-t)$
$subplot(2,2,1); plot(t, x); title('seno')$
$subplot(2,2,2); plot(t, y); title('coseno')$
$subplot(2,2,3); plot(t, z); title('exponencial')$
$subplot(2,2,4); plot(t, w); title('-exponencial')$



3-D:

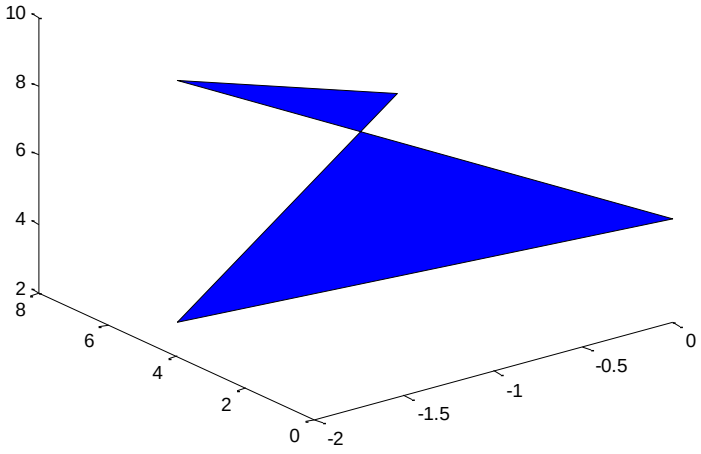
Matlab también nos permite graficar en 3D, para eso está el comando $plot3$, y a diferencia del comando $plot$, ahora se ingresara 3 datos.

$t = 0{:}\,360$
$x = \mathrm{sind}(t)$
$y = cosd(t)$
$plot3(t, x, y)$



Al igual que en 2D, en las gráficas en 3D también hay lo que es el comando $fill$, el cual se redefine por $fill3$. Por ejemplo:

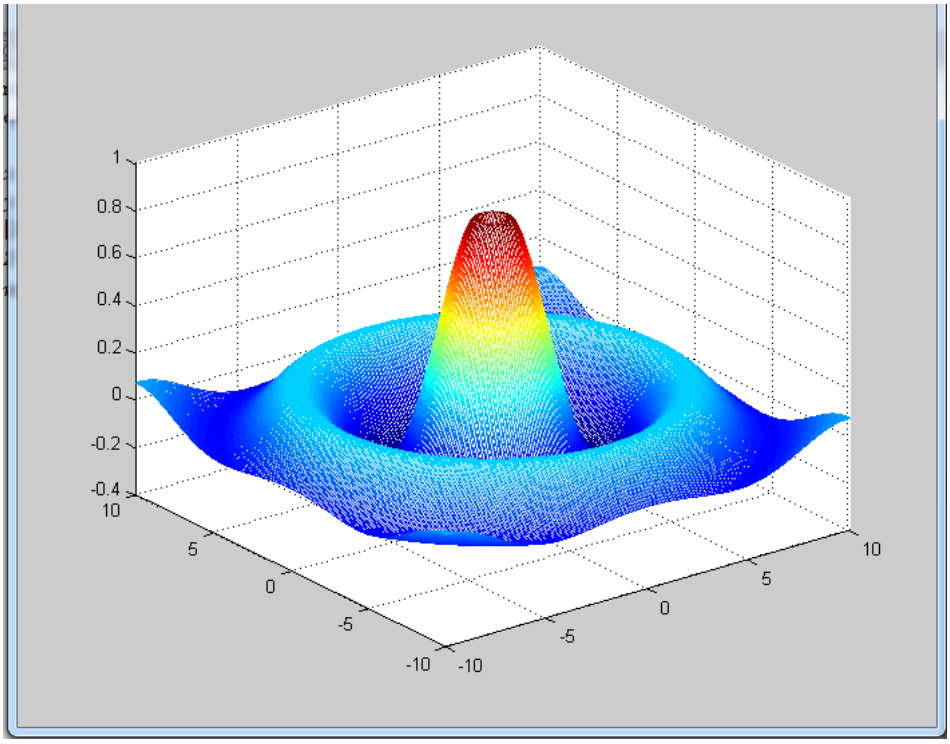
$x = [-2\ 0\ 2\ 0\ -2]$
$y = [4\ 8\ 4\ 0\ 4]$
$z = [3\ 5\ 10\ 5\ 3]$
$fill3(x, y, z, 'b')$



Superficie de Mallas:

Esto se puede graficar mediante el uso de los comandos $meshgrid$ (el cual crea dos matrices a partir de dos vectores) y el comando $mesh$ (el cual se encarga de graficar), por ejemplo:

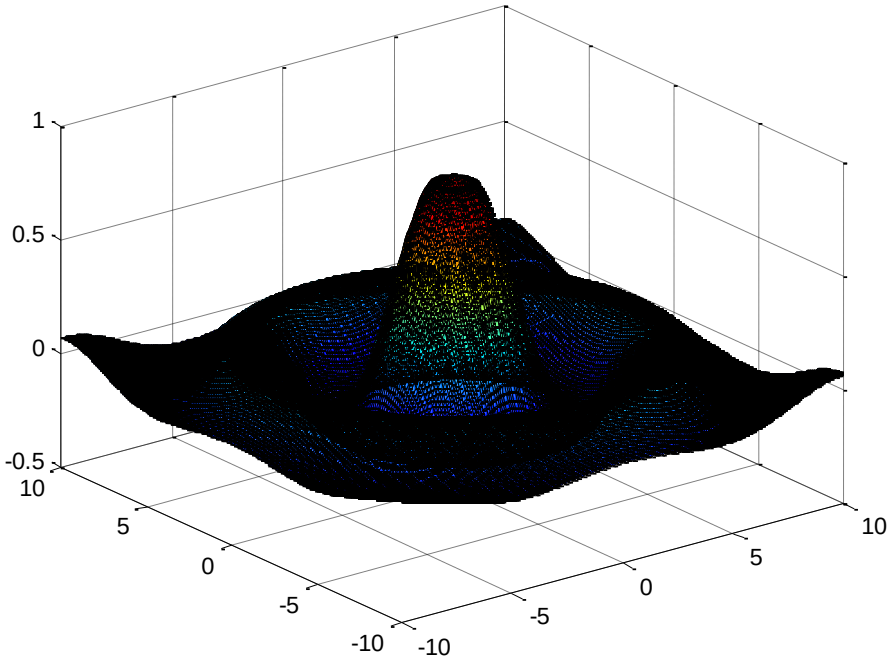
$x = -10{:}\,0.1{:}\,10$
$y = -10{:}\,0.1{:}\,10$
$[X\ Y] = meshgrid(x, y)$
$Z = \sin\big(sqrt(X.^2 + Y.^2)\big)./sqrt(X.^2 + Y.^2 + 0.1)$
$mesh(X, Y, Z)$



Se observa que la gráfica con el $mesh$, es simplemente una malla sin relleno, por lo cual, si se quiere dar color a esos espacios en blanco se hará el cambio de $mesh$ por $surf$, y se hará la misma estructura.

$x = -10{:}\,0.1{:}\,10$
$y = -10{:}\,0.1{:}\,10$
$[X\ Y] = meshgrid(x, y)$
$Z = \sin\big(sqrt(X.^2 + Y.^2)\big)./sqrt(X.^2 + Y.^2 + 0.1)$

$surf(X, Y, Z)$



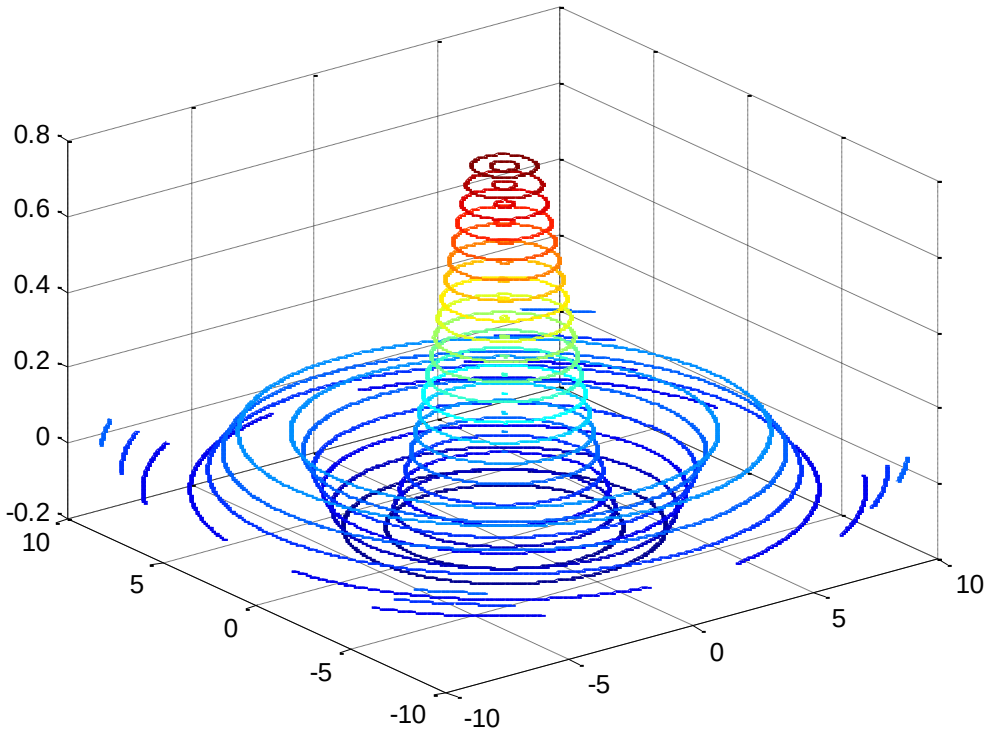
Superficies de Nivel:

Matlab también permite observar las superficies de nivel, esto con el comando $contour3$, lo cual tiene la siguiente sintaxis:

$$contour3(var1, var2, var3, \#\ de\ curvas)$$

Ejemplo:

$x = -10{:}\ 0.1{:}\ 10$
$y = -10{:}\ 0.1{:}\ 10$
$[X\ Y] = meshgrid(x, y)$
$Z = \sin\left(sqrt(X.^2 + Y.^2)\right)./sqrt(X.^2 + Y.^2 + 0.1)$
$contour3(X, Y, Z, 20)$



Ahora, si lo que deseamos ver son las curvas de nivel, es decir, en dos dimensiones, se cambia el comando $contour3$, por el comando $contour$.

Estadística descriptiva y análisis de datos

- Así como ya hemos visto la ayuda que brinda *Matlab* en gráficas y lo que es el álgebra lineal, está también tiene su espacio para la estadística, con distintas funciones o comandos.

Estadística descriptiva:

- **Media aritmética:**

$$a = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

El comando es $mean(a)$, $mean(a, n)$; $n = 1{,}2$

Si $n = 1$ (default), media de columnas
Si $n = 2$, media de filas

Ejemplos:

$a = [2\ 4\ 5\ 6\ 7];$
$mean(a)$

$b = [2\ 3\ 4; 5\ 3\ 2; 7\ 8\ 9]$
$mean(b) = mean(b, 1)$
$mean(b, 2)$

$a = linespace(0{,}3)$
$mean(a)$

$b = [3\ 3\ 2; 5\ 1\ 4]$
$mean(b)\ \wedge mean(b, 2)$

- **Media geométrica:**

$$g = (x_1\ x_2\ \ldots\ x_n)^{1/n}$$

El comando $geomean(a)$, el cual tiene la misma sintaxis que el $mean(a)$

$geomean(a)$
$geomean(a, 1)$
$geomean(a, 2)$

- **Media armónica:**

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{a_i}}$$

$$harmmean(a)$$

Funciona igual que el $mean$ y el $geomean$.

- $trimmean(x, y)$: Calcula la media ajustada de una muestra determinada; es decir, excluyen, donde

$$k = \frac{Ny}{200}$$

n: Cantidad de datos
y: Percentil

Sea $a = [1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10]$
$y = 20$
$trimmean(a, y) = 5.5$

Queda claro que este comando es muy útil cuando tenemos datos atípicos.

$b = [1\ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 4\ \ 5\ \ 6\ \ 8\ \ 9\ \ 1]$
$z = 10$
$trimmean(b, z)$

- $\min(b)$: devuelve los valores minimos de la muestra.
- $\max(b)$: devuelve los valores máximos de la muestra.

Cuando b es una matriz se puede expresar:
$min/\max(b) = min/\max(b, [\,], 1)$ %columnas
$min/\max(b, [\,], 2)$ %filas

- Mediana: Calcula la mediana de la muestra

$median(a)$
$median(a, \dim), \qquad dim = 1{,}2$

Medidas de dispersión:

Desviación estándar

Matlab toma dos clases de desviación estándar

- ✓ Desviación estándar corregida

$$\sigma = \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{1/2}$$

- ✓ Desviación estándar sin corregir

$$\sigma = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{1/2}$$

Desviación estándar corregida: $std(a)$

Desviación estándar sin corregir: $std(a,1) = std(a,0) = std(a)$

Si a es una matriz, se tiene

$$std(a, flag, dim)$$

Donde,

flag: $0 \vee 1$

dim: $1 \vee 2$

Varianza

- ✓ $var(a)$, donde a es un vector o matriz % columnas
- ✓ $var(a,1)$, varianza sin corregir

$(\text{std})^2 = \text{var}$, varianza corregida

Rango: Diferencia entre el vector máx y min de una serie de datos.

$range(a)$, a vector o matriz

Cuando a es matriz:

$range(a) = range(a,1)$ %columnas

$range(a,2)$ %filas

Rango intercuartil: Diferencia entre el percentil75 y el 25.

$iqr(a)$

$iqr(a) = iqr(a, 1)$

$iqr(a, 2)$

Percentil: Calcula el valor de un percentil determinado en el intervalo $[0\ 100]$ de una muestra especifica.

$prctile(a, p)$

a: muestra (vect o matriz)

p: Percentil buscado (escalar o vector)

a	p	$prctile(a, p)$
Vector	Escalar	Calcula el percentil p de la muestra a.
Matriz	Escalar	Genera un vector con el percentil p, de cada columna.
Vector	Vector	Genera un vector con los percentiles $p(i)$.
Matriz	Vector	Genera una matriz, en el cual cada columna corresponde a los percentiles especificados en p correspondiente a cada columna de la matriz.

Ejemplo:

$a = 1{:}10$
$b = [25\ \ 50\ \ 75]$
$prctile(a, b) = [3\ \ 5.5\ \ 8]$

$a = [1\ \ 2\ \ 3\ \ ;\ \ 7\ \ 5\ \ 6\ \ ;\ \ 4\ \ 5\ \ 6\ \ ;\ \ 8\ \ 9\ \ 1]$
$b = [25\ \ 50\ \ 75]$
$prctile(a, b) = [2.5\ \ 3.5\ \ 2\ \ ;\ \ 5.5\ \ 5\ \ 4.5\ \ ;\ \ 7.5\ \ 7\ \ 6]$

Oblicuidad de una muestra:
La oblicuidad es una medida de asimetría de las muestras con distribución normal, se mide a partir de la media. Este nos dice que si el valor es menor que cero, significa que la mayoría de datos se encuentran a la izquierda de la media, si es mayor que cero, están a la derecha, y si es cero, la muestra corresponde a una

distribución normal con perfecta simetría.

El comando para esto es $skewness(a)$, donde a puede ser vector o matriz.

- $skewness(a) = skewness(a, 1)$: lo cual está relacionado a las columnas de la matriz.
- $skewness(a, 2)$: lo cual está relacionado a las filas de la matriz.

Frecuencia absoluta: Cantidad de veces que se repite un dato.

$tabulte(a)$ Da una tabla donde se observa los distintos datos de la muestra "a", su frecuencia absoluta y el por ciento de los datos.

donde: a solo puede ser un vector.

$a = [4\ 1\ 4\ 4\ 2\ 3\ 4\ 3\ 1\ 2]$

$tabulate(a)$

Desviación absoluta media o mediana:

$$mad(a, b, c)$$

Donde a: vector o matriz

b: 0 (desviación media) ò 1 (desviación mediana)

c: 1 (%columnas) ò 2 (%filas)

Grupos de datos:

- **Covarianza:**

$cov(a) \Longrightarrow a$ vector, devuelve un valor.
$\Longrightarrow a$ matriz, cada fila es una observación y cada columna una variable.

- **Correlación lineal (Pearson):**

$corr(a)$: "a" es una matriz y da una matriz de correlación entre columnas.
$a = [1\ 2\ 3; 7\ 5\ 6; 8\ 9\ 1]$
$corr(a)$

$corrcoef(a)$: Devuelve una matriz con los coeficientes de correlación, en el cual las filas son observaciones y columnas variables. "a" es una matriz.
$corrcoef(a)$

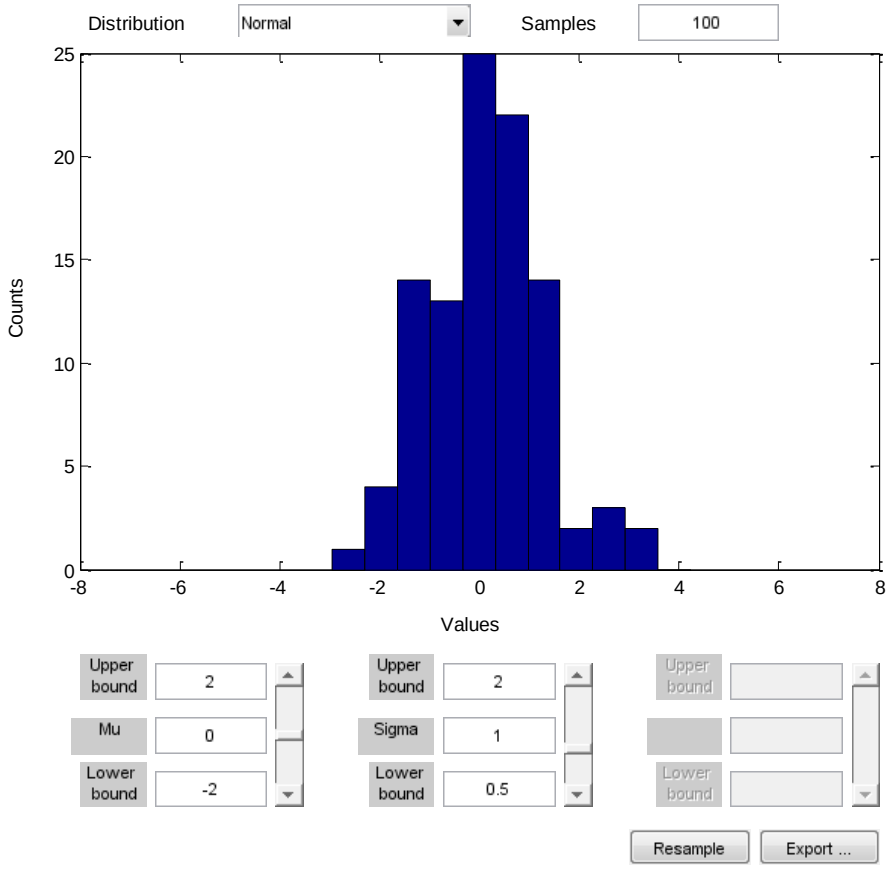
Graficas Estadísticas

Matlab tiene muchos comandos predeterminados para las gráficas estadísticas.

- **Randtool:**
  Esta función permite generar de forma interactiva números al azar mostrando los resultados gráficos por medio de un histograma.

  La interfaz que se abre con la función, permite fijar valores de parámetros para la distribución y para cambiar sus límites superiores e inferiores en la generación de datos aleatorios, e incluso permite cambiar la distribución.

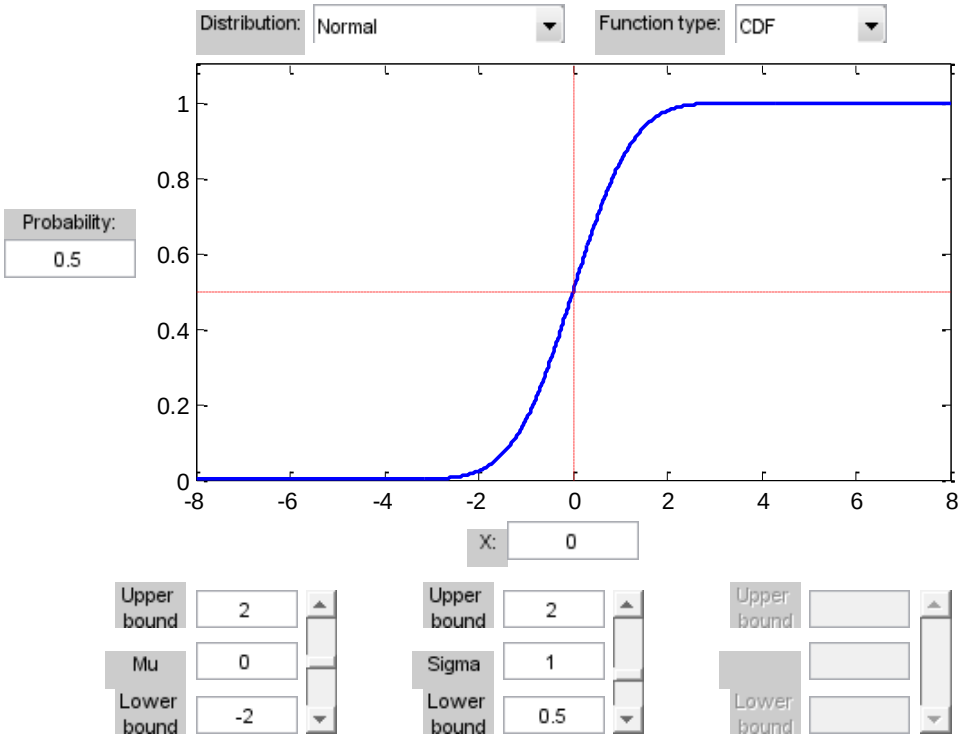
  $randtool$ ↲



- **Disttool**

  Esta función permite generar de forma interactiva diagramas de diferentes distribuciones de probabilidad. La interfaz generada por esta función permite escoger entre dos diagramas, el de cdf (genera una función distribución acumulada elegida) o el pdf (función de densidad de probabilidad para una distribución especifica).

  En la interfaz se tiene la posibilidad de conocer los valores de x, las cuales corresponden a un nivel de probabilidad, o viceversa.



- **Lsline**

  Esta función genera la línea de ajuste de los mínimos cuadrados de una función predeterminada.

  $x = randn(30,1)$ % vector con 30 datos aleatorios con distribución normal.
  $plot(x,' t')$
  $lsline$

- **Hist: Grafico de histograma**

  $hist(y)$: Grafico de histograma con diez barras para los valores contenidos en el vector y, y las barras están espaciadas entre el valor mínimo y máximo que toma la variable.

  $hist(y, nb)$: nb es el numero de barras.

  $y = normrnd(0,0.5,500,1)$ % la función $normrnd$ genera datos aleatorios
  %Donde $normrnd(\mu, digma, nº\ filas, nº\ columnas)$, distribución normal.
  $hist(y)$

Probabilidad

Distribucion binomial:

- Binofit( ): Estimacion del parámetro (x) o intervalos de confianza para datos de tipo binomial.
  $p = binofit(a, n)$; devuelve la máxima probabilidad para el suceso "a" en n oportunidades.

  Si a es un vector, entonces se devuelve un $p(i)$ para cada $a(i)$.

  Cuando n es un vector de igual dimensión que "a", se calcula $p(i)$ para cada $a(i)$ según $n(i)$.

  $p = binofit(2,5)$

  $p = 0.4$

  $a = [2,4,6,8]$

  $p1 = binofit(a, 8)$

  $p1 = [0.25\ \ 0.5\ \ 0.75\ \ 1]$

  $n = [4\ \ 8\ \ 12\ \ 16]$

  $p2 = binofit(a, n)$

  $p2 = [0.5\ \ 0.5\ \ 0.5\ \ 0.5]$

- Binocdf(): Función binomial de distribución acumulada

  $p = binocdf(x, n, p)$; donde (x, n, p) puede ser un vector o una matriz, sin embargo, deben tener iguales dimensiones.

  $p = binocdf(3,4,0.6)$
  $p = 0.8704$

Distribución de Poisson

Poissfit(): estimación del parámetro lambda o intervalo de confianza para datos que se acomodan a las condiciones de una Poisson.

$$\lambda = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$[lambda, intv] = poissfit(x, alpha)$

Genera el parámetro (λ) a partir de la muestra x.

-intev muestra un intervalo con $100(1 - alpha)\%$ de confianza, si no se especifica este parámetro el intervalo por defecto es de 95%.

$c = magic(3)$

$[lambda, intv] = poissfit(c)$

Poisscdf( ): Poisson acumulada.

$poisscdf(x, lambda)$; calcula el valor de la sumatoria de los valores Poisson para los respectivos parámetros, donde x puede ser un vector o una matriz, $\lambda > 0$.

Supongamos que en cierta área el nº x de tornados observados durante un año, tiene una distribución Poisson con $\lambda = 0$, entonces cual es la probabilidad de obtener:

- A lo mucho 5 tornados?

$$P(x \leq 5) \Longrightarrow a = poisscdf(5,8)$$

- Entre 6 y 9?

$$P(6 \leq x \leq 9) \Longrightarrow poisscdf(9,8) - poisscdf(6,8)$$

Integrales y Derivadas

Polinomios:

Comencemos hablando de polinomios, y cuál es la forma de ingresar estos a través de Matlab.

Sea:

$$p = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Se ingresa el polinomio anterior a Matlab de la siguiente forma:

$$p = [a_n \;\; a_{n-1} \;\; \ldots \;\; a_1 \;\; a_0]$$

Una vez que se ha ingresado un polinomio, otro tema de interés es saber como hallar sus raíces, lo cual se obtiene mediante el comando $roots$.

Ejemplos:

Sea:

$p = x^2 + 2x + 1$
$p = [1 \;\; 2 \;\; 1]$
$roots(p)$

$q = 3x^3 + 2x^2 - 1$
$q = [3 \;\; 2 \;\; 0 \;\; 1]$
$roots(q)$

Otro punto importante al trabajar con polinomios, es la evaluación de estos, para lo cual está el comando $polyval(p, x)$, donde p es el polinomio y x es el valor a evaluar, el cual puede ser escalar o vector.

Ejemplo:

$p = [3 \;\; 2 \;\; 0 \;\; 1]$
$x = 5$
$polyval(p, x)$

$p = [3\ \ 2\ \ 0\ \ 1]$
$x = [2\ \ 3\ \ 5]$
$polyval(p, x)$

Si tenemos dos polinomios y queremos dividirlos con la intención de tener una suma de fracciones parciales, Matlab también posee un comando para esto:

$$[r\ \ p\ \ k] = residue(b, a)$$

Donde:
b es el polinomio numerador.
a es el polinomio denominador.
r , p , k serán los vectores de salida y cumplirán lo siguiente.

Sea

$$\frac{b(x)}{a(x)} = \frac{b_m x^m + b_{m-1} x^{m-1} + \cdots + b_1 x + b_0}{a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0}$$
$$\frac{b(x)}{a(x)} = \frac{r_1}{x - p_1} + \frac{r_2}{x - p_2} + \cdots + \frac{r_n}{x - p_n} + k(x)$$

Ejemplo:

$$\frac{b(x)}{a(x)} = \frac{5x^3 + 3x^2 - 2x + 7}{-4x^2 + 8x + 3}$$

$b = [5\ \ 3\ \ -2\ \ 7]$
$a = [-4\ \ 8\ \ 3]$

$[r\ \ p\ \ k] = residue(b, a)$
$r = [-7.6736\ \ \ 0.7361]$
$\mathrm{p} = [2.3229\ \ -0.3229]$
$k = [-1.25\ \ \ -3.25]$

$$\frac{\mathrm{b(x)}}{\mathrm{a(x)}} = \frac{-7.6736}{\mathrm{x} - 2.3229} + \frac{0.7361}{\mathrm{x} + 0.3229} + (-1.25\mathrm{x} - 3.25)$$

Ahora hablaremos sobre derivadas, para lo cual Matlab también tiene un comando específico, cuando estamos en polinomios ingresados mediante vector. Este es $polyder$, y tiene la siguiente sintaxis.

$polyder(p)$, donde p es un polinomio.
$polyder(p, a)$, donde $p\ y\ a$ son polinomios y devuelve la derivada del producto de ellos.
$[q\ \ d] = polyder(a, b)$, donde $a\ y\ b$ son polinomios y devuelve la derivada del cociente.

Ejemplo:

$a = [3\ \ 6\ \ 9]$
$b = [1\ \ 2\ \ 0]$

$polyder(a) = [6\ \ 6]$

$polyder(a, b) = [12\ \ 36\ \ 42\ \ 18]$

$[q\ \ d] = polyder(a, b)$
$q = [-18\ \ -18]$
$d = [1\ \ 4\ \ 4\ \ 0\ \ 0]$

Otra de las cosas importantes en polinomios es el hecho de integrar, para lo cual esta el comando $polyint$, y tiene la siguiente sintaxis.

$polyint(p)$, donde p es el polinomio a integrar.
$polyint(p, k)$, donde p es el polinomio a integrar y k es la constante de integración.

Nota:

$$poly(p) = poly(p, 0)$$

Ejemplo:

$a = [3\ \ 6\ \ 9]$
$polyint(a) = [1\ \ 3\ \ 9\ \ 0]$

$k = 2$
$polyint(a, k) = [1\ \ 3\ \ 9\ \ 2]$

Funciones

Es claro que nosotros no trabajamos sólo con polinomios, es más, estos son una pequeña parte de todas las funciones a trabajar; sin embargo, para trabajar estas no se ingresan simplemente como vector, por lo cual ingresaremos un nuevo comando, $syms$, el cual crea variables simbólicas.

$$syms \;\; var_1 \, var_2 \ldots \;\; var_n$$

Ejemplo

$syms \;\; x \;\; y \;\; z$
$f = x + y$
$g = \sin(x)$
$h = [x \;\; y \;\; z]$

Es claro que las operaciones elementales no tiene la necesidad de algún comando.

Una vez que ya sabemos trabajar con variables simbólicas, comenzaremos a dar los comandos importantes para darle una mayor utilidad a esto, para lo cual empezaremos con el comando $subs$, el cual sirve para evaluar las variables.

Ejemplo

$syms \;\; x \;\; y$
$f = x^2 y$
$subs(f, 2) = 4y$
$subs(f, [x, y] , [2,3]) = 12$

Otro comando importante es el que nos ayudará a encontrar los ceros de nuestras funciones ingresadas, el cual se basa en el comando $solve(f, x)$, donde f será la función a ingresar y x será la variable con respecto a quien se quiere hallar los ceros.

Ejemplo

$syms \;\; x \;\; y$
$f = x^2 + 2x + 1$
$solve(f, x) = [-1 \;\; -1]$

$g = x + y$

$solve(g, x) = -y$
$solve(g, y) = -x$

Ahora hablaremos de otro comando importante, el cual nos ayudara a encontrar la derivada de una función ingresada, hablamos del comando $diff$, el cual tendrá la siguiente sintaxis:

- $diff(f)$: derivada de la función f con respecto a la variable x.
- $diff(f, v)$: derivada de la función f con respecto a la variable v.
- $diff(f, n)$: $n - esima$ derivada de la función f con respecto a la variable x.
- $diff(f, v, n)$: $n - esima$ derivada de la función f con respecto a la variable v.

Ejemplo:

$syms\ \ x\ \ y\ \ z$
$f = \text{z}\sin(x)\cos(y)$
$diff(f, z) = \sin(x)\cos(y)$
$diff(f, x, 3) = -z\ \sin(x)\cos(y)$

Y para terminar, hablaremos del comando para integrar, el cual queda expresado por int, y tiene la siguiente sintaxis:

- $int(f)$: integral de la función f con respecto a la variable x.
- $int(f, v)$: integral de la función f con respecto a la variable v.
- $int(f, v, a\ , b)$: integral de la función f con respecto a la variable v, con limites de integración a y b.

Ejemplo:

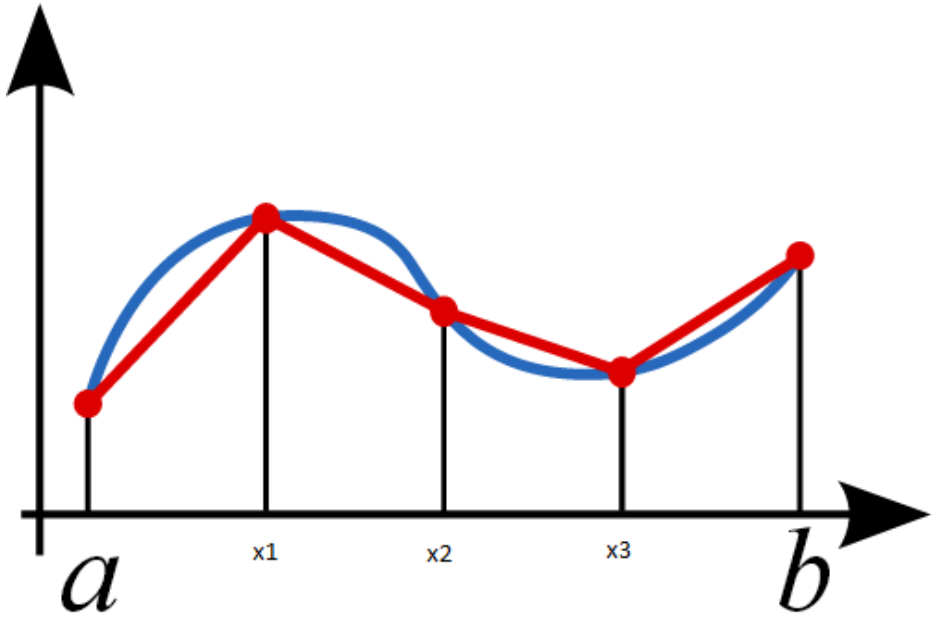
$syms\ \ x\ \ z$
$g = \dfrac{-2x}{1 + z^2}$
$int(g, x) = \dfrac{-2}{1 + z^2}$
$int(g, z, 4\ , 7) = -2x\ \text{atan}(0.1034)$

Métodos de Integración Numérica

Sabemos que no todas las funciones son posibles de integrar en forma analítica, por lo cual recurrimos a los métodos numéricos, con el fin de tener una solución a esto, no olvidando que la mencionada solución es tan solo una aproximación.

Regla del Trapecio:

Una forma de aproximar una integral definida, consiste en usar N trapecios, para lo cual debemos de tener que la función es continua y positiva sobre el intervalo de integración.



Se debe tener en cuenta que la distancia entre los puntos es la misma, y en este caso, para cuatro trapecios se tiene:

$$\int_a^b f(x)dx \approx \frac{b-a}{8}(f(b)+2f(x3)+2f(x2)+2f(x1)+f(a))$$

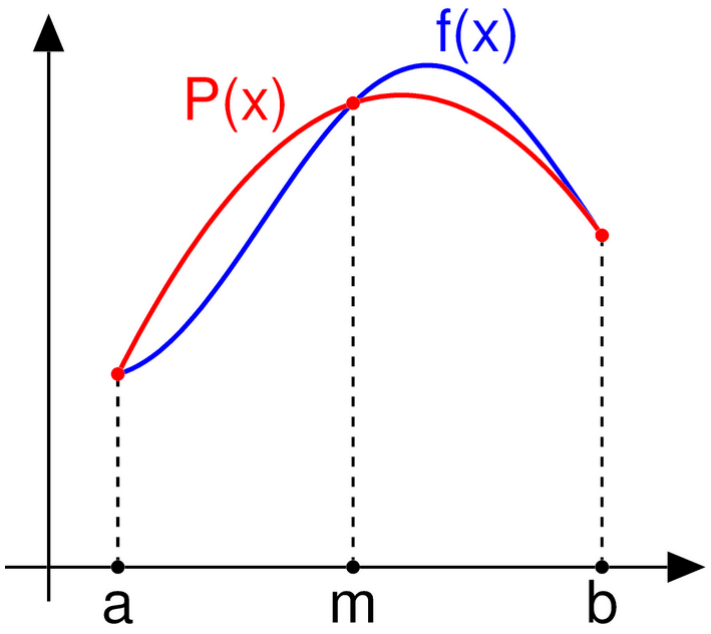
Y en forma general, para una cantidad N de trapecios, se tendrá:

$$\int_a^b f(x)dx \approx \frac{b-a}{2N}(f(b)+2f(x_{n-1})+\cdots+2f(x_1)+f(a))$$

Regla de Simpson 1/3:

En comparación con la regla del trapecio, la regla de Simpson 1/3 nos dará una aproximación mas precisa del valor de la integral buscada.

La idea para la construcción de este método es conectar tres puntos de la función a integrar mediante una parábola, y la integral viene a darse por el área que se forma bajo la parábola. Cabe mencionar que la parábola buscada se forma a través del polinomio interpolante de Lagrange.



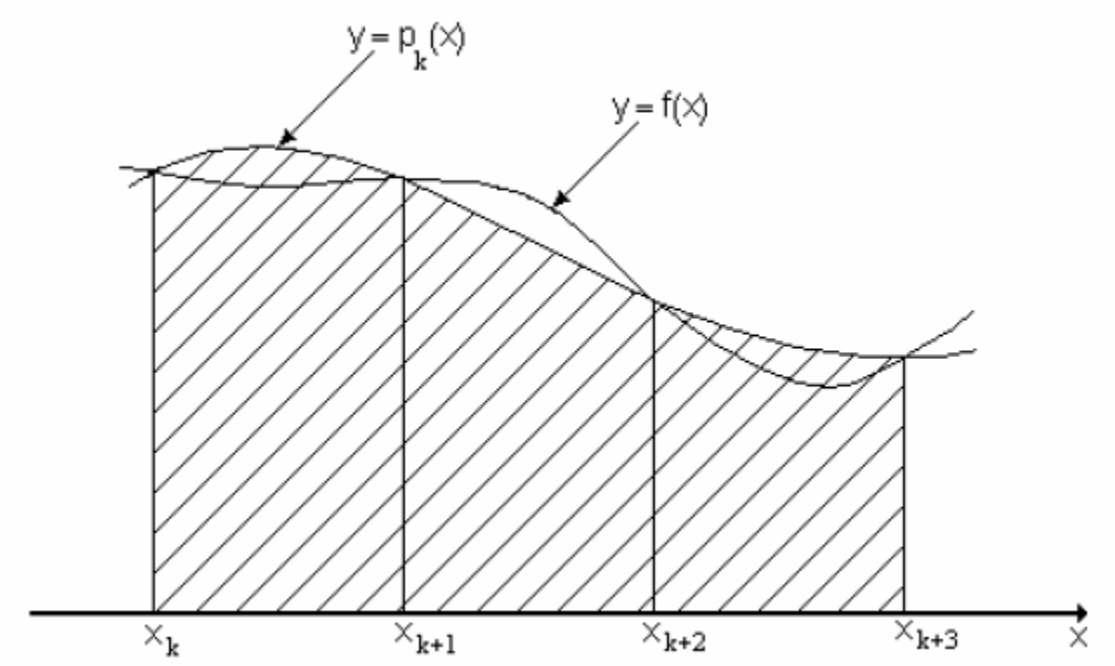
Otra cosa importante a tomar en cuenta es que el punto m mostrado, es el punto medio del intervalo de integración $[a, b]$.

Luego, la aproximación del valor de la integral queda expresada de la siguiente forma:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}(f(b) + 2f(m) + f(a))$$

Regla de Simpson 3/8:

Este método guarda una estrecha relación con la regla de Simpson 1/3, con la única e importante diferencia de que para la construcción del polinomio, se usan cuatro puntos, con lo cual se obtendrá un polinomio de grado 3, y luego de esto, análogo a lo anterior, se hallara el área formada bajo el polinomio encontrado.



Y la formula general viene expresada de la siguiente forma:

$$\int_{x_k}^{x_{k+3}} f(x)dx \approx \frac{x_{k+3} - x_k}{8}\left(f(x_{k+3}) + 3f(x_{k+2}) + 3f(x_{k+1}) + f(x_k)\right)$$